

Dr. Dobb's Journal

For Users of Small Computer Systems

G OPERATION DESCRIPTION G O

3	A <u>v</u> , <u>i</u>	Add <u>i</u> to <u>v</u>	3	N
1	Aa <u>v</u>	Add <u>v</u> to <u>a</u> (<u>v</u> ≠ 1t)	6	N
	ABab	Add BCD <u>lb</u> to <u>la</u>		N
	ABtu	Add BCD <u>lMDu</u> to <u>lM</u>		
4	AMa <u>v</u>	Add <u>a</u> to <u>v</u> (memor)		
4	ASd <u>v2</u>	Arith. shift <u>v2</u>		
	ASd <u>ma</u> , <u>sc</u>	Arith. shift <u>ma</u>		
1	At <u>vx</u>	Add <u>vx</u> to <u>t</u>		
	AXmab	Add extended <u>mb</u>		
	AXmtu	" " <u>mMDu</u> to <u>mMD</u>		
2	Ar <u>v</u>	Arith. shift <u>v</u> (memor)		
B	af <u>i2</u>	Test bit <u>i2</u> of <u>a</u>		
	fb	" " <u>b</u> "		
	<u>i2</u>	Test bit <u>i2</u> of		
	<u>y</u>	Test bit <u>y</u> (in		
	<u>i2</u>	Branch to		
	<u>y</u>	Test bit <u>i2</u> of		
	<u>y</u>	" " <u>y</u> (in		
	<u>y</u>	Branch to		
5		Test bit		
5		Test b		
3		Com		
1				
6				

$\tan(x) = \sin(x) \cdot 1000$

$\cos(x)$

934614

1 2 3 4 5 6 7 8 9 0

Q W E R T Y U I O P

A S D F G H J K L

— Z X C V B N M . ' —

**Portable Pidgin
for the Z80**

**68000 Cross
Assembler**

**8086 Graphics
Subroutines**

and even more!

**includes the DDJ
CP/M Exchange!**

Z-80[®] and 8086 FORTH

PC/FORTH[™] for IBM[®] Personal Computer available now!

FORTH Application Development Systems include interpreter/compiler with virtual memory management, assembler, full screen editor, decompiler, demonstration programs, utilities, and 130 page manual. Standard random access disk files used for screen storage. Extensions provided for access to all operating system functions.

Z-80 FORTH for CP/M [®] 2.2 or MP/M	\$ 50.00
8086 FORTH for CP/M-86	\$100.00
PC/FORTH for IBM Personal Computer	\$100.00

Extension Packages for FORTH systems

Software floating point	\$100.00
Intel 8087 support (PC/FORTH, 8086 FORTH only)	\$100.00
AMD 9511 support (Z-80, 8086 FORTH only)	\$100.00
Color graphics (PC/FORTH only)	\$100.00
Data base management	\$200.00

Nautilus Cross-Compiler allows you to expand or modify the FORTH nucleus, recompile on a host computer for a different target computer, generate headerless code, and generate ROMable code with initialized variables. Supports forward referencing to any word or label. Produces load map, list of unresolved symbols, and executable image in RAM or disk file. No license fee for applications created with the Cross-Compiler! Prerequisite: one of the application development systems above for your host computer.

Hosts: Z-80 (CP/M 2.2 or MP/M), 8086/88 (CP/M-86), IBM PC (PC/DOS or CP/M-86)

Targets: Z-80, 8080, 8086/88, IBM PC, 6502, LSI-11

Cross-Compiler for one host and one target	\$300.00
Each additional target	\$100.00

FORTH Programming Aids by Curry Associates. Includes Translator, Callfinder, Decompiler, and Subroutine Decompiler. 40 page manual. Used with Cross-Compiler to generate minimum size target applications.

Specify host system	\$150.00
---------------------------	----------

Z-80 Machine Tests Memory, disk, console, and printer tests with all source code in standard Zilog mnemonics	\$ 50.00
---	----------

All software distributed on eight inch single density soft sector diskettes, except PC/FORTH on 5¼ inch soft sector single sided double density diskettes. Micropolis and North Star disk formats available at \$10.00 additional charge.

Prices include shipping by UPS or first class mail within USA and Canada. Overseas orders add US\$10.00 per package for air mail. California residents add appropriate sales tax. Purchase orders accepted at our discretion. No credit card orders.

Laboratory Microsystems

4147 Beethoven Street
Los Angeles, CA 90066
(213) 306-7412

Z-80 is a registered trademark of Zilog, Inc.

CP/M is a registered trademark of Digital Research, Inc.

IBM is a registered trademark of International Business Machines Corp.

Announcing the best Error Free Personal Computer Diskette Money can Buy. For Less.



- Error Free
- 1 year warranty
- Hub ring installed
- Write/Protect notch
- Next day delivery

\$19.90/box of 10

- No minimum order quantity
- 8" or 5 $\frac{1}{4}$ "
- Plus 2 Free BONUS DISKS/Box*

*5 $\frac{1}{4}$ " SPECIAL ONLY

If you are a member of a user group or a school district please call for special terms on future offers. **TSS** is the largest specialty supplier of magnetic media in the Midwest. We have the products that you want when you need them. Please take advantage of this introductory offer and call us **now**.



Transaction
Storage Systems, Inc.
MAGNETIC MEDIA SPECIALISTS

CALL TOLL FREE

1-800-521-5700

1-800-482-4770 (Michigan)

313-557-3036 (Detroit)

312-922-0076 (Chicago)

614-221-1788 (Columbus)

513-621-1518 (Cincinnati)

Telex 810-224-4646

EXPECT A MIRACLE

YES, **TSS** is the magnetic media supplier that I have always wanted.

Please send me _____, 8" ☐ 5 $\frac{1}{4}$ " ☐, @ _____ ea.
Box Quantity Price

I am interested. Please send me more information
or call me at () _____

For faster order entry call any of our toll-free or local numbers

Company _____

Name _____ Title _____

Address _____

City _____ State _____ Zip _____

Amex/Master Card/Visa orders are accepted Expiration Date

month _____ year _____

Circle no. 223 on reader service card.

October 1982 Volume 7, Issue 10

Editor — Marlin Ouverson

Assistant Editor — Fritz Lareau

Contributing Editors —

Dave Caulkins, Dave Cortesi,

Ray Duncan, Gene Head,

Michael Wiesenberger

Marketing Director — Craig Harper

Marketing Assistant — Beatrice Blatteis

Advertising Director — Laird Foshay

Circulation Manager — Gloria Romanoff

Circulation Assistants —

Terri Marty, Linda Marohn

Art Director — Clifford West

Production Manager — Barbara Ruzgerian

Production Assistant — Jane A. McKean

Typesetter — Renny Wiggins

Cover Illustration — Al McCahon

© 1982 by People's Computer Company
unless otherwise noted on specific articles.
All rights reserved.

Subscription Rates: \$25 per year within the United States; \$44 for first class to Canada and Mexico; \$62 for airmail to other countries. Payment must be in U.S. Dollars, drawn on a U.S. Bank.

Writer's Guidelines: All items should be typed, double-spaced on white paper. Listings should be produced by the computer, using a *fresh, dark ribbon* on continuous white paper. Please avoid printing on perforations. Payment is in contributor's copies. Requests to review galleys must accompany the manuscript when it is first submitted. Authors may receive a copy of the complete writer's guidelines by sending a self-addressed, stamped envelope.

Donating Subscribers Contributing Subscriber: \$50/year (\$25 tax deductible). **Retaining Subscriber:** \$75/year (\$50 tax deductible). **Sustaining Subscriber:** \$100/year (\$75 tax deductible). **Lifetime Subscriber:** \$1000 (\$800 tax deductible). **Corporate Subscriber:** \$500/year (\$400 tax deductible, receives five one-year subscriptions).

Contributing Subscribers: Thomas H. Grohne, G. V. Elkins, Robert M. Connors, William J. McEown, Robert C. Luckey, Robert N. Stabler, DeWitt S. Brown, John B. Palmer, Burks A. Smith, Transdata Corporation, Mark Ketter, John W. Campbell, Friden Mailing Equipment, Howard E. Decker, Jim Wilson, Frank Lawyer, Chris Pettus, Paul Lutus, Ben Goldfarb, Rodney Black, Steven Weiss, Pat Phelan, John Brodie, James Fleming, Unison World, Thomas Davis, Jan Steinman, Ronald E. Johnson, G. Hunter, Kenneth Drexler, Real Paquin, Ed Malin, John Saylor, Jr. **Sustaining Subscribers:** Pete Roberts, Steven Fisher Enterprises. **Lifetime Subscriber:** Michael S. Zick.

Foreign Distributors UK & Europe: Hofacker-Verlag, Tegernseer Strass 18, D-8150 Holzkirchen, West Germany. **Asia & Australia:** ASCII Publishing Co., Ltd, Aoyama Building 5F, 5-16-1 Minami Aoyama, Minato-ku, Tokyo 107, Japan.

CONTENTS

Articles

12 68000 Cross Assembler

by Allen Kossow

DDJ #68 described a homebrew, multi-68000 development system; readers responded with many requests for publication of the Fortran cross assembler mentioned in that article. The main program and subroutines will be presented in their entirety, beginning this month.

25 The Portable Pidgin

Z80 Macro-Assembly Implementation

by Herbert Gintis

Software portability often comes at the expense of size and speed. Author Gintis found Pidgin to be low-level, structured and portable. Here he presents his Z80 macro-assembly implementation intended to run under CP/M.

36 Simplified 68000 Mnemonics

by W. D. Maurer

Motorola's powerful and versatile 68000 microprocessor offers many features to the experienced programmer. But how does one ever become fluent in its complex assembler mnemonics? As this article points out, redesigning to a simpler mnemonic set may be a reasonable step to take.

Departments

6 Letters

6 Editorial

7 Dr. Dobb's Clinic

Debuggers, Nested SUBMITs and PC BASIC

42 CP/M Exchange

RCP/Ms, MBOOT3 and Downloading MODEM

51 16-Bit Software Toolbox

8086/88 Trig Lookup Functions

55 Of Interest

62 Advertisers Index

Dr. Dobb's Journal (USPS 307690) is published twelve times per year by People's Computer Company, Box E, 1263 El Camino Real, Menlo Park, CA 94025. Second class postage paid at Menlo Park, California 94025 and additional entry points. Address correction requested. Postmaster: send form 3579 to Box E, Menlo Park, California 94025 • 415/323-3111

QUNIX™

AT LAST—Multi-User and Multi-Tasking for the IBM PC

QUNIX is the total business, scientific, educational, and graphic Operating System for the IBM Personal Computer. Now you can unleash the full power of the IBM Personal Computer. The total capability of your computer is now available for future growth.

QUNIX—For the first time, the tremendous power of the QUNIX single-user single-tasking operating system is now available with MULTI-USER and MULTI-TASKING for up to 8 simultaneous users and up to 250 simultaneous tasks.

QUNIX—Now allows high speed arithmetic, high speed and high resolution graphics by supporting the 8087 coprocessor for number crunching.

QUNIX—The ideal business operating system designed for transaction and word processing that can be shared by up to 8 simultaneous users. No additional computer required. . . just add keyboards and terminals to the basic system.

QUNIX—Gives you mainframe/mini computer Data Processing capabilities. Features such as Hierarchical Files, Password Protection, 16-Character File and Directory Names, Data Communications to all mainframe, minicomputer and microcomputers, File Security and Winchester Support are all built into the fabulous QUNIX operating system.

QUNIX—After the frustration and limitations of DOS and CP/M™, you will LOVE QUNIX.



QUANTUM SOFTWARE SYSTEMS, INC.

7219 Shea Court
San Jose, California 95139
Tel: (408) 629-9402



QUANTUM SOFTWARE SYSTEMS, LTD.

P.O. Box 5318, Station "F"
Ottawa, Ontario, Canada K2C 3H5
Tel: (613) 235-4297



PC/SOFTWARE, INC.

926 Natoma Street,
San Francisco, CA 94103
(415) 863-0652

I am a ☐ User ☐ Dealer ☐ Distributor
☐ Please send me a brochure

Name _____ Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Computer _____ Disk Format _____

Dear *Dr. Dobb's*,

Recently, Digital Research ran ads in a newspaper in this area for people to work on CP/M for the Motorola 68000. Are we in for yet another copy of CP/M-80? That would be rather like running a Ferrari engine on one cylinder. (It is gratifying, however, that Digital Research has finally recognized that someone besides

Sincerely,
Jim Howell
5472 Playa Del Rey
San Jose, CA 95123

Dear Marlin Ouverson.

Enclosed is an article and program submitted for your review and inclusion in *Dr. Dobbs's*, if you so desire. Let me add that I enjoy your magazine very much, have all the issues from #1, and it

I have modified the Small-C compiler to work on a 370 under VM. How's that for reverse engineering? It also has FOR, GOTO, two-dimension arrays, etc.

Regards,
Chris L. Torkildson
13791 Heywood Ct.
Apple Valley, MN 55124

I have been using Small-C since the Software Works first offered it for CP/M. I recently got enough memory to allow it to compile itself correctly, and have started implementing all the changes suggested

(Continued on page 59)

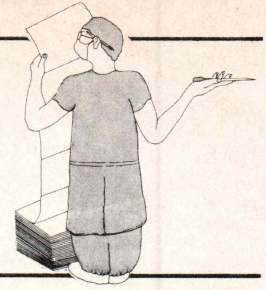
Going Public . . .

Many generous and sympathetic authors should be thanked for making *DDJ*'s mission successful. In addition to our present contributors, we remember the work of Dennis Allison, Steve Wozniak, Gary Kildall, Tom Pittman,

In the interest of building the public domain, we encourage authors not to reserve rights to programs published in *DDJ*, even though we will publish ones which reserve commercial rights. If the technical growth potential of computers is going to be realized, it won't be by those large companies with heavy investments in current technology — it will be by individuals working together to push beyond the limits of today's product line. When you hear one of us refer to the *DDJ community*, that's what we mean.

— Marlin Ouerson

by Dave Cortesi



The Very Last SUBMIT Item

For months we have been droning on about the SUBMIT command of CP/M, telling you how useful it can be and showing how to increase its usefulness with fixes and small utilities. In June we showed the utilities PAUSE and BEEP, which increase the operator's control over a submitted job. In August we brought you Robert Pasky's nicely-integrated set of patches that make SUBMIT handle lowercase input, control characters and null lines correctly. With them applied, you can submit command streams to ED and PIP for unattended execution. In September we presented Don Wright's QUITIF program, which gives the designer of a SUBMIT file the ability to check for some errors.

Now, courtesy of Digital Research, we SUBMIT users have the ultimate in SUBMIT patches. The July/August issue of *Microsystems* reproduces, unedited and without commentary, a set of applications notes, the work of the Technical Support Group at Digital Research, Inc. One of these is a lengthy patch which causes SUBMIT to append its output to an existing \$\$\$SUB file, instead of replacing that file. The result is that we can include a SUBMIT command in a submitted job. The inner submit file will be appended to the active submit file; when it completes, the outer submit file will be resumed with the line following the SUBMIT command.

Nested submits! And they come so easily, too. When SUBMIT runs, it places the submitted commands, one per 128-byte record, in reverse order in the file \$\$\$SUB. This was originally done, we think, for the convenience of the Console Command Processor (CCP). The CCP consumes records from \$\$\$SUB from the end of the file toward the beginning. That lets it use the "record count" byte of the directory entry for \$\$\$SUB as its pointer to the next record of the file. It couldn't keep such a pointer in storage, because there is no location in storage that is proof against overlay by an application program.

But this use of \$\$\$SUB makes the file work like a push-down stack. The CCP pops records off the end of the file. The new patch makes SUBMIT push records onto the end of the file, rather than erasing the file and building a whole new one. Nesting of submitted jobs follows automatically.

The Digital Research patch, with our own comments, is shown in Listing 1 (page 58). Assemble it, then use the I and R commands of DDT to overlay its hex file onto a copy of SUBMIT.COM (one with the Pasky Patches applied to it). The updated SUBMIT command does just about everything we could ask of it, so with this item we are turning the whole matter of CP/M SUBMIT over to Gene Head and the *CP/M Exchange*.

Debugging the Debuggers

We've had some letters from people who have problems with the Digital Research debug utilities, DDT and ZSID. L. Barker sent us a note on two holes in ZSID. It doesn't interpret the LDIR instruction correctly when the operands (the two strings defined by the HL, DE and BC registers) overlap on each other. The LDIR (long move) instruction of the Z80 copies a source string into a target string. When the target string overlaps on the source string (as when the source begins at 0200h, the target at 0201h, and the length is 255 bytes), LDIR becomes a Replicate instruction, duplicating the leading byte of the source into the target over and over. ZSID doesn't handle it right, though; it stops after copying one byte.

Barker also noted that the disassembler in ZSID can't seem to handle the "LD A,R" and "LD R,A" instructions. Its List command won't display them and its Assemble command won't accept them.

Gavin Brickell of Auckland, New Zealand, thought that he'd found a bug in DDT, but he hadn't. It appeared that DDT was modifying a word in storage without any reason. However, it had a reason: the word in question was the top word of the stack as defined by the program under test. DDT uses one word on the program's stack in at least three points during the execution of the Go command. The remarkable thing is that DDT manages to get by with only one word of the program's stack; it seems not to need any more.

Dana Trout of Goleta, CA, wrote in with a method for changing the number of the RST vector that DDT uses for breakpoints. One of the Digital Research applications notes mentioned above covers the same area. We think that DDT's use of RST 7 should be left alone if at all possible. DDT (and SID, and ZSID, and any other debugging tool) is going to use

some restart vector, and RST 7 is the one that is documented as being for this use. The great majority of CP/M systems will leave RST 7 free for debuggers. A modified debugger, exported to another system, is quite likely to crash that system by conflicting with a legitimate interrupt vector. Furthermore, there's a good reason for using RST 7 for debug breakpoints: most hardware will return FFh when asked to read from RAM that doesn't exist. FFh is the opcode for an RST 7 instruction; if that is the breakpoint vector, a wild branch to non-existent memory will cause an automatic breakpoint!

Finally, Nick Hammond of Pebble Beach, CA, contributed a cute trick that can be played with DDT. DDT could be useful for poking around in the innards of CP/M, if only it would leave things alone. "The problem with DDT," Hammond writes, "is that it changes things when it gets control. Location 5 no longer points to the BDOS except via a circuitous series of jumps, and the CCP is missing, having

CP/M Software

UNERA

Recovers all ERASED Files for CP/M 2.2 Floppy and Hard Disk Systems with standard directories. Also CP/M 1.4 with 8" Single Density Directory. Includes patching information for CP/M 1.4 Double Density.

.....\$75.00

FORMS-3

Microsoft programs for filling out forms. Features field editing, justification, multi-pages, required entry. Can also use a separate data file.

.....\$40.00

SUPERFILE

Solves your filing problems. Menu driven information retrieval system for storing and quickly finding information. Features AND, OR and NOT in search command. Sort, merge and split utilities included. Build data base with any CP/M Editor.

with Demo Data Base & Manual \$195
Manual only (applies to purchase) \$50

CP/MUG.DB

Data base of catalogs and abstracts of 80+ disks from the CP/M users group library. Use with Superfile.

.....\$25.00

Superfile and CP/MUG.DB ...\$210.00
AVAILABLE 8" SINGLE DENSITY, NORTH STAR
SINGLE AND DOUBLE DENSITY.

ADD \$1.50 SHIPPING AND HANDLING

Elliam Associates

24000 Bessemer Street
Woodland Hills, CA 91367
(213) 348-4278



Compare



Victor's Desktop Business Computer System.

Businesses today face a basic dilemma when it comes to selecting a computer.

So-called "personal" computers have limited power and capacity. They're just too small to be useful to most businesses.

And the larger mini computers are more expensive.

Victor has a solution to that dilemma.

The Victor 9000 Business Computer is priced under \$5,000. Like a "personal."

Yet the Victor 9000 has a capacity that rivals the expensive minis.

A close look at the chart shows you just how the Victor 9000 compares.

The Victor gives you the kind of memory and storage capacity business applications demand. Much more than the IBM Per-

sonal Computer, the Apple III or their competitors.

For more information, call Victor at (800) VIC-9000.

Or Write Victor Business Products, P.O. Box 1135, Glenview, IL 60025

VICTOR BUSINESS PRODUCTS
Subsidiary of Kidde, Inc.
KIDDE

MAKE & MODEL	Victor 9000	IBM PC	Xerox 820	Apple III	Radio Shack TRS80 Model II
Processor Type	8088	8088	Z80A	6502	Z80A
Word Length	16 bits	16 bits	8 bits	8 bits	8 bits
Memory Size (Internal)	128-896KB	16-256KB	64KB	96-256KB	32-64KB
Storage Capacity on 2 Floppies	2400KB (5 1/4")	640KB (5 1/4")	160KB (5 1/4")	280KB (5 1/4")	960KB (8")
CRT Display					
Standard Format	80 x 25	80 x 25	80 x 24	80 x 24	80 x 24
Alternate Format	132 x 50	None	None	None	None
Graphics Resolution	800 x 400	640 x 200	None	560 x 192	None
Communications					
Built-in Serial Ports at no extra cost	2	0	2	1	2
Built-in Parallel Ports at no extra cost	1	0	2	0	1
Human Factors					
Keys on Keyboards	94-104	83	96	74	76
Detached Keyboard mechanism	Yes	Yes	Yes	No	Yes
Tilting Display mechanism	Yes	No	No	No	No
Swivelling Display	Yes	No	No	No	No
Desk Area Required (Approx. Square In. with 2 floppy disks)	310	420	470	361	500
Operating System	CP/M-86*			Apple DOS	TRS DOS
Supplied Standard	MS-DOS	None	None		

NOTE: Chart based on manufacturer's information available as of April 4, 1982.

*CP/M is a registered trademark of Digital Research, Inc.

VICTOR®

Serving American business for 65 years.

Circle no. 276 on reader service card.

been overwritten." Hammond supplies a technique for getting a copy of DDT into the transient area below the CCP, and getting control into it with the BDOS vector and the CCP intact. It works like this.

First, invoke DDT as a command, with its own .COM file as argument:

```
A>ddt ddt.com
```

DDT will start up and load DDT.COM. Give the Go command with no address; control will enter the loaded copy. It will relocate itself just below the first-started copy of DDT, clear of the CCP, put its own vector into the RST 7 location, and prompt for a command. Use the Assemble or Substitute command to put an RST 7 instruction at location 0100h. Then use the G0 or a control-c to do a warm start. The CCP will be reloaded over the first copy of DDT, but the second copy will remain. Then create and call a null command file as a means of causing a branch to location 100h, where you stored a

RST 7:

```
A>save 0 null.com
```

```
A>null
```

Control will pass to the RST 7 at location 100, which will return control to the second copy of DDT. You can now use it to peer about the system, observing low storage and the CCP in their native states.

Ask Uncle

"I have a Diablo 1620," says Ernest Knipp of Houston, TX, "and the Courier-10 printwheel has a pound-sterling symbol. I can't get this symbol to print. Is there any way to reach this unreachable character?"

You betcha, Ernest. There are 96 spokes on the Diablo printwheel. Ninety-four of them print in response to the 94 printing characters of the ASCII code. Two of the spokes correspond to the ASCII space (32h) and the DEL (7Fh) codes. Those two can't be reached by sending the printer a single ASCII code.

When it gets a space, the printer just advances the carriage; it doesn't use the printwheel at all. And it conforms to the ASCII standard by ignoring DEL characters entirely.

However, the manual for our 1650 printer says that the sequence ESCape, "Y" will print the spoke matching 32h, and ESCape, "Z" will print the one matching 7Fh. In other words, if your printer receives the two bytes 1Bh, 59h, it should print the sterling symbol. If it receives the bytes 1Bh, 5Ah, it should print a logical-not symbol.

Assorted Grumps

A person who would rather remain anonymous has told us that he had it from an equally anonymous, but supposedly well-informed, source that the code of the IBM PC's BASIC interpreter was indeed produced by an 8080-to-8086 translator program. That's all very

(Continued on page 58)

NOW—A COMPLETE CP/M PASCAL—FOR ONLY

\$29.95!

Goodbye BASIC, PL/1, COBOL—hello PASCAL! Now, to make this most advanced language available to more micro users, we've cut our price—to an amazing **\$29.95!** This astonishing price includes the complete JRT Pascal system on diskette and the comprehensive new user manual. Not a subset, it's a *complete Pascal for CP/M.** Check the features:

- Separate compilation of external procedures
- Auto-loading
- 14 digit FLOATING POINT arithmetic
- True dynamic storage
- Verbal error messages
- Fast one-step compiler: no link needed
- Graphing procedures
- Statistic procedures
- Activity analyzer
- Prints program use histogram
- Advanced assembler interface

THIS IS THE SAME SYSTEM WE SOLD FOR \$295!

So how can we make this offer?—why the unbelievable deal? Very simply, we think all software is overpriced. We want to build volume with the booming CP/M market, and our overhead is low, so we're passing the savings on to you.

AND AT NO RISK!

When you receive JRT Pascal, look it over, check it out. We invite you to compare it with other systems costing ten times as much. If you're not completely satisfied, return the system—with the sealed diskette unopened—within 30 days and your money will be refunded in full! **THAT'S RIGHT—COMPLETE SATISFACTION GUARANTEED OR YOUR MONEY BACK!**

In addition, if you want to copy the diskette or manual—so long as it's not for resale—it's o.k. with us. Pass it on to your friends! **BUT ACT TODAY—DON'T DELAY ENJOYING PASCAL'S ADVANTAGES—AT \$29.95, THERE'S NO REASON TO WAIT!**



To: **JRT SYSTEMS**
1891—23rd Avenue
San Francisco, CA 94122
phone 415/566-4240



O.K. You've sold me. Send me JRT Pascal by return mail. I understand that if I'm not completely satisfied, I can return it within 30 days—with the sealed diskette unopened—for a full refund.
I need ☐ 8" SSSD diskette. ☐ 5 1/4" diskette for ☐ Northstar, ☐ Osborne, ☐ Apple-CP/M, ☐ Heath, ☐ Superbrain.

Name _____ Address _____

City _____ State _____ Zip _____

☐ Check ☐ C.O.D. ☐ Mastercharge ☐ VISA
(CA residents add sales tax. Add \$6 for shipping outside North America.)

Card # _____ Exp. _____

Signature _____
*CP/M is a Digital Research TM. A 56K CP/M system is required.

COMPUVIEW'S CP/M-86 GIVES YOU WHAT IBM CAN'T

Increased Productivity

Improve your productivity with built-in horizontal scrolling (254 columns) and screen line editing. This lets you extensively edit or re-enter any command line on the screen and greatly reduces the amount of re-typing necessary due to mis-typed or repeated commands. It's almost like having a built-in full screen editor for every program you use. And with 25% more disk capacity you will be swapping disks a lot less.

We Don't Lock You In

Read and write not only IBM CP/M-86 disks, but also IBM MSDOS and other CP/M double density disks. Transfer files with other CP/M and CP/M-86 computers via the serial port. The screen driver with status line emulates many popular terminals. And of course we're software compatible with IBM.

Winchester Disk Support

Special versions available to support the TECMAR, DeVong, Corvus and other hard disks. Or have a quad density 96 tpi double sided floppy as drive B (or C and D externally) with 782K capacity.

No Software Shortage

Most CBASIC programs run perfectly with our CP/M-86 and CBASIC-86. Even most programs compiled with CBASIC 8080 will run with CBASIC-86. And Pascal-MT is available too.

V-COM DISASSEMBLER Labels, ASCII, Exceptional Speed

No other Z80 CP/M disassembler produces understandable source code as quickly as V-COM. It is INTEL and ZILOG compatible, and features easy to read code with a cross reference table. Best of all, it can create source code with user defined labels, storage areas and ASCII strings. V-COM is exceptionally fast and can disassemble a typical 12K .COM file into a 76K .ASM file, containing 7500 lines of source code, and a 33K cross reference file in under two minutes with 8" SD floppies. (About five times faster than others).

You can create two auxiliary files to easily specify labels for 8 and 16 bit values and the location of storage areas, tables and ASCII strings. The disassembled code can be sent to the console, the disk and the printer, or any combination at once.

Each package includes a 30 page manual, sample program files and variations of V-COM compatible with TDL, MAC and

Compare CompuView with IBM CP/M-86

Feature	Compuview	IBM
Horizontal Scrolling	Yes	No
Screen Line Editing	Yes	No
Page Control	Yes	No
Emulate Popular Terminals	Yes	No
'Smart' CRT Functions	Yes	No
Read/Write IBM MSDOS Disks	Yes	No
Serial File Transfer	Yes	No
Support Non-IBM Hardware	Yes	No
Menu Driven Configuration	Yes	No
Programmable Function Keys	Yes	Yes
Status Line	Yes	Yes
Serial and Parallel Printers	Yes	Yes
File Capacity	193K	154K

CP/M-86 for IBM PC\$285
Quad Density Drive Version ..\$350
Winchester Disk Version\$425
Manual Only\$20

VEDIT-86 with above purchase.
This version of VEDIT has horizontal scrolling (254 columns) ..\$125

CBASIC-86\$325
PASCAL-MT-86\$600

Tandon double sided 80 track drive gives 782K file capacity. Fits into IBM PC as drive B, or connect two externally. CompuView CP/M-86 required.\$450

More Storage Lower Cost Than IBM

IBM Double Sided
40 Track Drive\$650
IBM CP/M-86\$280
Total (320K file capacity) ...\$930

Tandon Double Sided
80 Track Drive\$450
CompuView CP/M-86\$350
Total (782K file capacity) ...\$800

ZILOG assemblers. Feature for feature, no other disassembler at any price even comes close.\$80
Manual only\$12

8086 SOFTWARE

NEW Terminal and File Transfer program for IBM PC, Displaywriter and other CP/M-86 and MSDOS systems. (Superset of MODEM7)\$70

VEDIT full screen editor for CP/M-86, MSDOS, IBM Personal Computer and IBM Displaywriter.\$195

CP/M-86 BIOS for popular S-100 disk controllers and SCP 8086 computer. Source Code.\$90

Bootable CP/M-86 disks for popular S-100 computers ..Call



Sophisticated program development editing with useful word processing features and powerful TECO commands and macros.

Performance

Fast and easy 'What you see is what you get' full screen editing of files up to one disk in length. Completely replaces ED and reduces your editing time by 90%. Includes search and replace, text move and copy, complete file handling and printing. Insert a specified line range of another file anywhere in the text, and even change disks while editing. Unique automatic indenting for use with Pascal, PL/I, 'C' and others. For assembly language, VEDIT can change lower case letters typed in the label, opcode and operand fields to upper case, and leave the comments unchanged. VEDIT's small 13K size allows up to 45K of a large file to be memory resident. Even if you already have a good word processor, VEDIT's program development features, powerful command structure, and exceptional speed will greatly increase your productivity.

User Oriented

Fully adapts to your hardware and applications with a menu driven customization for keyboard layout, CRT selection, any screen size, tab positions, default editing parameters and more. Includes a status line with the cursor's line and column positions, an 'Undo' key and recovery from full disk conditions (you can delete files or change disks). VEDIT is significantly enhanced twice a year and you can opt for our popular and inexpensive update option. And we offer direct technical support you can count on, including custom patches for new CRT terminals.

Text Buffers, Macros

Ten buffers can hold text, macro commands or complete files. These may be edited, loaded and saved on disk. Macros perform repeated or complex editing operations. (Perform 40 search/replace on 20 files automatically, for example.) Buffers allow extensive 'cut and paste', including portions from multiple files. Buffers can be preloaded at startup.

Word Processing

Includes word wrap, adjustable margins, reformatting of paragraphs, word and paragraph functions, and printing with imbedded printer control characters. May be used stand-alone or in conjunction with most text output processors.

Hardware Support

CRT version supports over 40 terminals including the VT-100. Utilizes 'smart' terminal capabilities for fast screen updating. Your keyboard layout can use any function and cursor keys. Memory mapped version offers high speed, flexibility and supports bank select (SSM VB3). Startup command file can initialize programmable function keys, VEDIT parameters, and more.

Ordering

Please specify your microcomputer, video board or the CRT terminal version, 8080, Z80 or 8086 code and disk format.

VEDIT - Disk and Manual	
For 8080 or Z80	\$150
For CP/M-86 or MSDOS	\$195
Manual only	\$18

Xerox 820 ● Apple II Softcard ● TRS-80 II and I
SuperBrain ● Zenith Z89 ● DEC VT180 ● Televideo
Northstar ● Cromemco ● Altos ● Vector ● Micropolis
MP/M ● CP/M-86 ● MP/M-86 ● MSDOS

IBM Personal Computer and IBM Displaywriter

VISA and MASTERCARD

CP/M, CP/M-86 and MP/M are registered trademarks of Digital Research, Inc. Apple II is a registered trademark of Apple Computer, Inc. SoftCard is a trademark of Microsoft, Inc. TRS-80 is a trademark of Tandy Corp. IBM Personal Computer and Displaywriter are trademarks of International Business Machines, Inc.

1955 Pauline Blvd., Suite 200
Ann Arbor, Michigan 48103
(313) 996-1299

CompuView

PRODUCTS, INC.

68000 Cross Assembler

The June 1982 issue of *DDJ* contained a brief description of the computer system which is being developed by myself, Darryl Uchitil, and Jim Hannas as a test bed for system software and hardware using the Motorola 68000 micro-processor. The reader response to both *DDJ* and myself as a result of that article has been interesting and informative to Jim, Darryl and myself, and I would like to thank all of you who have called or written me since June. I will present at the end of this article a summary of answers to some of the most frequently asked questions about our system for the benefit of those who have not tried to contact me or who have written but have not yet received my reply.

Cross-Assembler Description

This article will describe the cross assembler that was mentioned in the June 1982 issue of *DDJ*. It was developed because we needed a way to write small programs for execution on a single 68000 processor to test the hardware and to

by Al Kossow

Allen Kossow, Medical College of Wisconsin, Department of Physiology, 8701 Watertown Plank Road, Milwaukee, WI 53226.

THE FORTH SOURCE™

Exclusively FORTH! Mountain View Press has the greatest selection of FORTH books, manuals, and disk programs available. From TRS80 and APPLE to PDP-11 and NOVA this is your FORTH source. Over 50 FORTH products are currently in stock and more are being added each week. Contact the FORTH source — Mountain View Press — for the latest in FORTH materials.

MOUNTAIN VIEW PRESS
PO Box 4656
Mountain View, CA 94040
(415) 961-4103

test the interface between the 68000 processors and the I/O bus control processor. The cross assembler for the 68000 has been run on two different PDP-11 systems, one running RT-11, and one running RSX-11M. It has also been run on a VAX/VMS system under the applications migration executive. It should be possible to adapt the assembler to other machines by recoding assembly-language subroutines for 32-bit arithmetic and modifying the routines that reference octal constants, but as of this writing, no attempt has been made to do so.

The cross assembler consists of a main program and a collection of subroutines, each performing part of the assembly process. The cross assembler is a two-pass assembler; the source code is read twice by the program. The first pass through the source code picks up all of the labels, and the second pass generates the listing and object files. All operations to files are handled by individual subroutines. These subroutines are SOURCE, LIST, and OBJECT. All machine-dependent routines for opening and closing of files are located in these routines.

Each of the three routines for the source, listing and object files is called when the program is first started, to determine where each is to be sent. Some checking is done to determine if just a carriage-return was typed, so that there is a default of only a listing to the terminal and no object file is established. Once the source for the assembler and the destinations of the listing and object are established, the assembler starts the assembly by setting the pass number to one and calls the subroutine PARSE.

PARSE scans the input line and determines if the statement is just a comment. If it is not a comment, it splits apart the label, opcode and operand fields of the statement, and returns pointers to these fields to be used by the statement evaluation routines.

After the line is parsed, the statement opcode is evaluated by the subroutine PROCESS. PROCESS calls DECOPC, which takes the characters pointed to by the opcode pointer from PARSE and attempts to find an opcode that matches. If an opcode is found, the values of one or more skeletal opcodes (opcodes without effective addresses or sizes) are returned.

After evaluating the opcode, the operands are evaluated to determine their general type. All "simple" operands, such as registers, are evaluated immediately, while "complex" operands (i.e., operands containing labels) are not evaluated.

If a valid opcode skeleton was returned, there will be a number returned by DECOPC which represents a general way in which to evaluate this type of operation. This number is used by PROCESS in a multi-way branch to different sections of code for evaluation of the opcode. Once in the specific section of code for the opcode, the operands are checked for validity in the opcode, and if the operand was "complex" it is evaluated and a value is returned.

Each specific section of code for an opcode builds up the opcode skeleton, filling in the size and effective address fields as necessary for that type of instruction. The result of this evaluation is an array of 16-bit values which represent the result of that line of the assembly, and a count of the number of words generated by that line.

At the end of the opcode processing routine is the section of code that handles labels. If a label was detected and is valid for the opcode type, the current value of the location counter is placed in the symbol table entry for that label.

If this is the second pass, the object and listing is generated based on the values in the instruction word array; otherwise, the next line is fetched and the evaluation process is repeated. After the last line of the program is read, the input file is closed, and a routine is called to print the contents of the symbol table.

Frequently Asked Questions About The Multi-68000 System

Q: Can I get a copy of your assembler?

A: If you don't feel like typing in the source code in this article, I can send you the source as it appears here on an 8" single-sided, single-density floppy disk in either CP/M or RT-11 format for \$25 on an as-is, no-support basis. I can provide the source on a CP/M disk, but I have no way to compile it or test it there.

Q: What is CLICS? Where can I get it?

A: CLICS is a collection of subroutines for 2-D graphics developed by Mike Garrett while he was with the Department of Defense. Since the time I wrote the article, I have heard that Mike left DOD and started a company selling graphics software written in C. I have not been able to reach him to determine if this is true or not. The CLICS software I have been working with came off of the 1980 Spring DECUS RSX-11 special interest group symposium tape. This tape is available from the DECUS library.

Q: When/where can I buy one of your systems?

A: The notion of selling this system has been discussed a number of times between the three of us, but we just don't want to get into the computer business. The schematics and blank PC boards will probably be available from us on an as-is, no-support basis, along with the software we develop, but we have no plans to sell or support systems.

Q: Why didn't you build your system on a "XYZ" bus?

A: I had expected much more flack about using the PUNIBUS than I actually received. The primary reason for not using a commercial bus was the desire to build a system which had a few large (8" by 15") cards to minimize the number of cards to build. We also probably would not have used a special bus if we had planned on selling the system once we finished it.

In closing, I would like to thank all of you that have taken the time to call or write me about our project, and will promise to keep you informed of our progress in future issues of *DDJ*.

DDJ

(Listing begins on page 14)

Reader Ballot

Vote for your favorite feature/article in each issue.
Circle Reader Service No. 520

We Speak

MANX[®]

software systems

Your Language...

AZTEC C II

An Outstanding C Compiler with full floating point.

Manx[®] Software Systems provides professional quality C Compilers that are used to implement language processors, business, scientific, statistical, word processing and general utility applications.

A full feature professional C Compiler for CP/M, Zenith HDOS and Apple (DOS or CP/M)

**static double pi = 3.1415926535898;
C = 2.0*pi*r;**

Manx has an outstanding selection of UNIX compatible C Compilers.

- Aztec C II CP/M \$199
- Aztec C II CP/M for the Apple \$199
(requires Z-80 card, language card and lower case)
- Aztec C][for Apple DOS \$199
- Aztec C II for Zenith HDOS \$199
- Aztec C for CP/M or HDOS \$145
- * C86 for CP/M-86 \$249
- * C86 for IBM PC DOS (MSDOS) \$249
(* by Computer Innovations)

Order today for prompt delivery

MANX[®]
software systems

**Box 55, Shrewsbury, N.J. 07701
(201) 780-4004**

Mastercard and Visa accepted

68000 Cross Assembler (XASM)

(Text begins on page 12)

Due to its length, this listing will be continued next month.

Listing One

PROGRAM MAIN
C
C M68000 CROSS-ASSEMBLER MAIN PROGRAM
C
C
C

C DATE: FEBRUARY 10, 1981
C
C

C REVISION: X1.0 (EXPERIMENTAL PRE-RELEASE)
C
C

C AUTHOR: Allen Kosow
C Medical College of Wisconsin
C Department of Physiology
C 8701 Watertown Plank Rd.
C Milwaukee, WI 53226
C

PGM NOTES:

C Symbols are unique to 8 chrs.
C Symbol table size is currently set to 512.
C Symbol table size is determined by the memory constraints
C of the machine it's running on. The current symbol table
C size works under RT-11 with 64kb of memory and under RSX-11.
C
C

C This program follows most of the conventions of the
C Motorola 68000 assembler with the following exceptions:

- 1) no macros or pc relative addressing (except branches)
- 2) generalized (i.e., simple) object file format
- 3) listing format similar to that of MACRO-11
- 4) doesn't handle adr regs as special cases (i.e., MOVEA is just a MOVE).

LOGICAL UNIT DEFINITION

- 1 = SOURCE FILE
- 2 = OBJECT FILE
- 3 = LIST FILE
- 5 = KEYBD

IMPLICIT INTEGER (A-Z)
BYTE NAME(8),SYNFB(513)

(Continued on column 2)

C	COMMON /LST/ UNIT,PASS,NAME,NOPAGE,NOLINE,MEFLG,IERCNT	C	C,....	READ ONE LINE OF SOURCE FILE
C	COMMON /SYMT/STIND,SYMAJR,PC,MOSYM,NEWPC,SYNFB	C	15	CALL IACLR(NEWPC)
C	COMMON /PRSE/ OPPTTR,MODPTR,OPMPTTR,LABEL,CMTPTR	C		CALL SOURCE(2)
C	+PRFLG,SCANPT,OPCLEN,OPMPT2,INODE	C	C,....	IF EOF DETECTED DO PASS 2
C	COMMON /SRC/ LNELEN,ISERR,NOCARD,SRCLNE	C		IF(ISERR,EQ,1) GOTO 20
C	COMMON /OBJOUT/ OBJBUF,OBJINC,LFLG,RFLG,DBFLG	C	C,....	RESET MULTIPLE ERROR FLG
C	COMMON /OPCPTS/ OPTYP,OPSKEL,OPSK2,OPIDX	C		MEFLG = 0
C	COMMON /CWMT / WORD,PL	C	C,....	PARSE SOURCE LINE
C	COMMON /HEXFLG/ ENDFLG,HEXMC,HEXPC,OLDPC	C		CALL PARSE
C	DIMENSION OBJBUF(40)	C	C,....	IF NULL LINE GET NEXT LINE
C	INTEGER#4 PC,NEWPC,SYMAJR(512),HEXPC,OLDPC	C		IF(PRFLG,EQ,0) GOTO 15
C	BYTE ERR,SRCLNE(81),LABEL(8),PL(132)	C	C,....	PROCESS SOURCE LINE
C	TELL FORTRAN TO IGNORE INTEGER OVERFLOWS ON	C		CALL PROCESS
C	MULTIPLY AND DIVIDE.	C		
C	ERR=128	C	C,....	IF END DETECTED DO PASS 2 ELSE GET NEXT LINE
C	THE FOLLOWING CALL IS NO 'ED OUT FOR FAP	C		IF(ISERR,EQ,1) GOTO 20
C	CALL SETERR(1,ERR)	C		I=JADD(PC,NEWPC,PC)
C	INITIALIZE VARIABLES	C		GOTO 15
C	NOPAGE=0	C	C,....	DO PASS 2
C	RFLG=1	C		REW SOURCE SET TO PASS 2 AND RESET PC
C	LFLG=1	C		CALL SOURCE(3)
C	OPEN FILES	C	20	PASS=2
C	CALL SOURCE(1)	C		IERCNT = 0
C	CALL LIST(1)	C		CALL IACLR(PC)
C	CALL OBJECT(1)	C		CALL IACLR(HEXPC)
C	DO PASS 1	C		CALL IACLR(OLDPC)
C	MOSYM=0	C	C,....	FLUSH PRINT BUFFER IN CASE ANYTHING LEFT
C	PASS=1	C		FROM LAST ASSEMBLY
C	CALL IACLR(PC)	C		DO 25 I=1,132
C	DO 10 I=1,8	C	25	PL(I) = *40
C	NAME(1)=32	C		INITIALIZE OBJECT BUFFER
C	CONTINUE	C	C,....	
C		C		

(Continued on column 3)

(Continued on page 16, column 1)

HAZELWOOD COMPUTER SYSTEMS

907 E. Terra Lane, O'Fallon, MO 63366 (314) 281-1055

256K DYNAMIC MEMORY

The DM-256 is a 256K dynamic memory for S-50C and S-64 systems. Based on the same design as the field proven DM-64, it utilizes the latest advances in memory technology to bring new economy in memories for larger systems.

- Greater than 2 MHz operation
- Fully transparent refresh
- 24 bit addressing
- Vertical or horizontal access

Order: DM-256 Price: \$1595

64K DYNAMIC MEMORY

The DM-64 is a 64K dynamic memory capable of fully transparent operation at bus speeds greater than 2 MHz. This proven memory design offers the economy of dynamic memory with the reliability of static memory.

- Greater than 2 MHz operation
- Fully transparent refresh
- 20 bit addressing
- 4K segment disable

Order: DM-64 Price: \$495

DMA FLOPPY DISK CONTROLLER

The DD-5/8 is a double side, double density DMA disk controller for 5 and 8 inch floppy disk drives.

- WD 1797 controller
- DMA operation at 2 MHz
- Controls 8 drives
- Fully automatic motor control
- Independent 5 and 8 sections
- Arbitration for multiple DMA
- 24 bit addressing
- Drive ready interrupt

Order: DD-5 (5 inch) Price: \$395
DD-8 (8 inch) \$395
DD-5/8 (both) \$450

HELIX™ MAINFRAME

The HM-64 is the mainframe of the HELIX computer system. It offers the S-64 bus, a compatible superset of the S-50C bus. Its ferro-resonant power transformer provides protection from power line voltage variations and noise. The attractive and functional cabinet includes space and power for 2 5-inch disk drives.

- 10 main bus slots (64 pin)
- 14 I/O bus slots (30 pin)
- S-50C compatible
- 2 MHz operation
- integral serial ports (2)
- integral parallel ports (2)
- 16, 20, or 24 bit I/O decoding
- Modular adaptable I/O panel
- Faraday shielded bus lines
- 35 Amp power supply
- Board access without extender
- Structurally solid

Order: HM-64 Price: \$1250
Options: Rack mount, blank panel,
230 V 50 Hz

6809 PROCESSOR

The CP-09 utilizes a 2MHz processor and supporting logic to offer the most advanced 6809 processor board available for the S-50C bus.

- 2 MHz operation
- 6 ROM positions (2 sets of 3)
- Software selection of ROMs
- Battery backup clock/calendar
- Programmable timer
- 1K RAM
- Compatible DAT REGISTERS
- Hardware breakpoint
- Console connector

Order: CP-09 Price: \$450

I/O CONTROLLERS

VC-256 256x256
B/W Graphics \$350
SC-2 2 Port Sync/Async
Comm \$295
BC-4 4 Port Buffered Async \$395

68000 PROCESSOR

The CP-68K employs an 8 MHz 68000 processor to bring 32 bit performance to the S-50C bus. When used with the S-64 bus and appropriate memory, full 16 bit data transfers are possible.

- 8 MHz processor
- 24 bit addressing
- 8/16 bit data transfers
- 4K ROM and 2K RAM
- Integral "Base-Bound" MMU

Order: CP-68K Price: \$995

PACKAGED SYSTEMS

The following HELIX systems are available at special package prices:

64K 6809 HELIX	\$1995
256K 6809 HELIX	\$2995
512K 6809 HELIX	\$4450
64K 68000 HELIX	\$2595
256K 68000 HELIX	\$3695
512K 68000 HELIX	\$4995

5 inch floppy disk subsystems:
(includes cable and controller)

40 track single side	\$1095
40 track double side	\$1295
80 track double side	\$1695

SOFTWARE

HLXBUG 6809 Monitor ROM (includes FLEX drivers)	\$50
OS-9 Level One	\$200
OS-9 Level Two	\$500

All items available stock to 30 days and are shipped assembled, tested, and burned in.

Hazelwood Computer Systems reserves the right to change prices and/or specifications without prior notice.

Available at authorized HELIX dealers

HELIX is a trademark of Hazelwood Computer Systems
FLEX is a trademark of Technical Systems Consultants
OS-9 is a trademark of Microware

Dealer inquiries invited. We support our dealers.

68000 Cross Assembler (XASM)

(Listing continued, text begins on page 12)

Listing One

```
ENDFLG = 0
HEXMC = 0
```

```
C PRINT FIRST PAGE HEADING
```

```
C CALL NEWPAG
```

```
30 CALL IACLR(NEWPC)
```

```
ORJMC = 0
```

```
CALL SOURCE(2)
```

```
C EOF DETECTED
```

```
C IF(ISERR.EQ.1) GOTO 50
```

```
C RESET MULTIPLE ERROR FLG
```

```
C MEFLG = 0
```

```
C PARSE LINE
```

```
C CALL PARSE
```

```
C PRINT A LINE OF ONLY COMMENTS NORMALLY
```

```
C IF(CMPTTR.EQ.1) GOTO 40
```

```
C CHECK FOR PARSING ERRORS
```

```
C IF(PRFLG.EQ.0) GOTO 30
```

```
C PROCESS IT
```

```
38 CALL PROCESS
```

```
C GENERATE LISTING
```

```
40 CALL LSTLINE
```

```
C CHECK IF THERE IS OBJ CODE TO GENERATE
```

```
C IF(OBJMC.EQ.0) GOTO 45
```

```
CALL BLDOBJ
```

```
C DO NEXT LINE IF NOT END
```

```
45 IF(ISERR.EQ.1) GOTO 50
```

```
I=JADD(PC,NEWPC,PC)
```

```
GOTO 30
```

```
C END OF ASSEMBLY, OUTPUT BALANCE OF OBJ BUFFER
```

```

C
200 ISERR=0
    READ(1,210,END=250) (SRCLINE(I),I=1,80)
210 FORMAT(80A1)
    NOCARD=NOCARD+1
C
C CONVERT ALL CHARACTERS
C
    DO 225 I=1,80
    IF(SRCLINE(I).GE.32) GO TO 220
    SRCLINE(I)=32
215 GO TO 225
220 IF(SRCLINE(I).LT.96) GO TO 225
    SRCLINE(I)=SRCLINE(I)-32
    IF(SRCLINE(I).GE.96) GO TO 215
225 CONTINUE
C
C REMOVE TRAILING BLANKS
C
    LMLEN=80
230 IF(SRCLINE(LMLEN).NE.32) GO TO 240
    LMLEN=LMLEN-1
    IF(LMLEN.GT.0) GO TO 230
240 LMLEN=LMLEN+1
    SRCLINE(LMLEN)=0
    GO TO 500
C
C END OF FILE
C
250 ISERR=1
    GO TO 500
C
C REWIND SOURCE FILE
C
300 REWIND 1
    NOCARD=0
    GO TO 500
C
C CLOSE SOURCE FILE
C
400 CLOSE(UNIT=1)
500 RETURN
    END
SUBROUTINE LIST(LCODE)
C
C PERFORMS OPEN AND CLOSE ON LIST FILE
C
C INPUT:LCODE = 1 => OPEN FILE (NAME READ FROM KEYBOARD)
C           2 => CLOSE FILE
C
    BYTE FILNAM(12)
    INTEGER PASS
    BYTE NAME(8)
    COMMON /LST/ UNIT,PASS,NAME,NOPAGE,NOLINE,MEFLG,IERCNT
    COMMON /FNAM/ FILNAM,OBJFLG

```

```

C INPUT:
C ICODE = 1 => FIND OPERAND IN SYMBOL TABLE, IF NOT FOUND,
C IT IS ENTERED INTO THE TABLE AS REFERENCED
C BUT NOT DEFINED, THE INDEX OF THE SYMBOL
C IN THE SYMBOL IS RETURNED IN 'STIND'.
C
C 2 => FIND LABEL IN SYMBOL TABLE, IF FOUND AND ALREADY
C DEFINED AND THIS IS THE FIRST PASS OF THE
C ASSEMBLER, THE MULTIPLE DEFINED BIT IS SET IN
C SYMFLG. IF FOUND BUT ONLY PREVIOUSLY REFERENCED,
C THE DEFINED BUT PREVIOUSLY REFERENCED BIT IS SET
C AND THE REFERENCED BIT IS CLEARED, IF NOT FOUND,
C IT IS ENTERED AND THE DEFINED BIT IS SET.
C
C IADDR = ADDRESS OF SYMBOL FOR ENTERING INTO SYMBOL TABLE.
C SYMBOL = SYMBOL TO LOOK UP OR ENTER IN SYMBOL TABLE.
C
C OUTPUT:
C STIND = INDEX INTO SYMBOL TABLE FOR SYMBOL.
C
C FORMAT OF 'SYMFLG':
C
C BIT MEANING IF SET
C 0 SYMBOL HAS BEEN REFERENCED BUT NOT DEFINED.
C 1 SYMBOL HAS BEEN DEFINED AND WAS REFERENCED BEFORE DEFINITION.
C 2 SYMBOL HAS BEEN DEFINED AND THERE WERE NO REFERENCES BEFORE.
C 3 SYMBOL HAS BEEN MULTIPLE DEFINED.
C 4 SYMBOL IS AN EQUATED VALUE
C
C IMPLICIT INTEGER (A-Z)
C BYTE SYMFLG(513),SYNSTR(8),SRCLINE(81)
C DIMENSION SYNSTR(4),SYNSTR(4),SYNSTR(4),SYNSTR(512)
C INTEGER*4 SYMADR(512),IADDR
C INTEGER*4 PC,NEWPC
C COMMON/SYMT/STIND,SYMADR,PC,NOSYN,NEWPC,SYMFLG
C BYTE NAME(8)
C COMMON /LST/ UNIT,PASS,NAME,NOPAGE,NOLINE,MEFLG,IERCNT
C COMMON /SRC/ LMLEN,ISERR,NOCARD,SRCLINE
C COMMON/SYMH/SYHSTR,SYNSTR
C
C PACK SYMBOL TWO BYTES TO A WORD
C
    DO 100 J=1,4
    I = J*2
    SYMBOL(J) = ((SYHSTR(I-1)*256).OR.SYHSTR(I))
100
C
C SEARCH FOR SYMBOL IN SYMBOL TABLE
C
    STIND = 1
    MOVFLG = 0
    IF(NOSYN.EQ.0) GO TO 200
    DO 120 STIND=1,NOSYN
    DO 110 J=1,4
    IF(SYHSTR(J,STIND).NE.SYMBOL(J)) GO TO 115
110 CONTINUE
    GO TO 300
115 DO 118 J=1,4

```



```

C
50  ENDFLG = 1
    CALL BLDOBJ
C
C,... PRINT SYMBOL TABLE
C
C    CALL PST
C
C,... CLOSE FILES AND DO IT AGAIN
C
    CALL SOURCE(4)
    CALL LIST(2)
    CALL OBJECT(2)
    GOTO 5
    END
    SUBROUTINE SOURCE(ICODE)
C
C PERFORMS ALL OPERATIONS OF SOURCE INPUT FILE
C
C INPUT:
C ICODE = 1 => OPEN SOURCE FILE (NAME READ FROM KEYBOARD)
C        2 => READ ONE LINE FROM SOURCE FILE INTO
C            'SRCLINE' (8001 FORMAT), TRAILING BLANKS
C            ARE DELETED, ZERO CHAR IS INSERTED AT
C            THE END OF THE LINE.
C        3 => REWIND SOURCE FILE.
C        4 => CLOSE SOURCE FILE.
C
C OUTPUT:
C
C SRCLINE = SOURCE LINE FOR CODE 2
C LNLLEN = LENGTH OF LINE FOR CODE 2
C ISERR = 1 IF END OF FILE ON READ (ZERO OTHERWISE)
C NOCARD = CARD NUMBER READ FROM SOURCE (1-?)
C
    BYTE FILNAM(12)
    BYTE SRCLINE(81)
    COMMON /SRC/LNLLEN,ISERR,NOCARD,SRCLINE
    COMMON /FNM/ FILNAM,OBJFLG
C
C SELECT FUNCTION
C
    GO TO (100,200,300,400),ICODE
C
C OPEN SOURCE FILE
C
    100  TYPE 110
        110  FORMAT('$$$rc file name: ')
            READ (5,120) ICNT,FILNAM
        120  FORMAT(0,12A1)
            IF (ICNT.EQ.0) STOP
            CALL ASSIGN(1,FILNAM,ICNT)
            NOCARD=0
            GOTO 500
C
C READ SOURCE LINE

```

(Continued on page 16, column 2)

```

C
C SELECT FUNCTION
C
    GO TO (100,200),LCODE
C
C,... ASSIGN DEFAULT LISTING TO CONSOLE
C
    100  LUNIT=5
        TYPE 110
        110  FORMAT('$$$st file name: ')
            READ (5,115) ICNT,FILNAM
        115  FORMAT(0,12A1)
            IF (ICNT.EQ.0) GOTO 116
C
C,... IF THERE IS A FILENAME ASSIGN LISTING TO LUN 3
C
    116  LUNIT = 3
        CALL ASSIGN(LUNIT,FILNAM,ICNT)
        NOPAGE=0
        GO TO 300
C
C CLOSE FILE
C
    200  IF (LUNIT.EQ.5) RETURN
        CALL CLOSE(LUNIT)
        RETURN
        END
    SUBROUTINE OBJECT(ICODE)
C
C PERFORMS OPEN AND CLOSE ON OBJECT FILE
C
    BYTE FILNAM(12)
    INTEGER PASS
    BYTE NAME(8)
    COMMON /LST/ LUNIT,PASS,NAME,NOPAGE,NOLINE,MEFLG,IERCNT
    COMMON /FNM/ FILNAM,OBJFLG
    GOTO (100,200),ICODE
    100  TYPE 110
        110  FORMAT ('$$$bj file name: ')
            READ (5,115) ICNT,FILNAM
        115  FORMAT(0,12A1)
            IF (ICNT.EQ.0) GOTO 116
            CALL ASSIGN(2,FILNAM,ICNT)
            OBJFLG = 1
            RETURN
        116  OBJFLG = 0
            RETURN
        200  IF (OBJFLG.EQ.0) RETURN
            CALL CLOSE(2)
            RETURN
            END
    SUBROUTINE SYNTAX(ICODE,IADDR,SYNSTR)
C
C SYMBOL TABLE PROCESSOR
C

```

(Continued on page 16, column 3)

```

    IF (SYNSYN(J,STIND),LT,SYMBOL(J)) GOTO 120
    IF (SYNSYN(J,STIND).EQ.SYMBOL(J)) GOTO 118
    MOVFLG = 1
    GOTO 200
    118  CONTINUE
    120  CONTINUE
C
C SYMBOL WAS NOT FOUND
C
    200  IF (NDSYN.LT.513) GO TO 210
        CALL ERROR(221)
        STIND=513
        GOTO 400
    210  IF (MOVFLG.EQ.0) GOTO 218
        ITEM = NDSYN
        DO 212 J=1,4
        212  SYNSYN(J,ITEM+1) = SYNSYN(J,ITEM)
            CALL JMOV (SYNAADR(ITEM),SYNAADR(ITEM+1))
            SYNFLG(ITEM+1) = SYNFLG(ITEM)
            SYNLIN(ITEM+1) = SYNLIN(ITEM)
            ITEM = ITEM + 1
        IF (ITEM.GE.STIND) GOTO 211
        NDSYN = NDSYN + 1
        DO 220 J = 1,4
        220  SYNSYN (J,STIND) = SYMBOL(J)
            IF (ICODE.EQ.1) GO TO 250
            SYNFLG(STIND)=4
            CALL IACLR(SYNAADR(STIND))
            I=IADDR(SYNAADR(STIND),IADDR,SYNAADR(STIND))
            SYNLIN(STIND) = NOCARD
            GOTO 400
    250  CALL IACLR(SYNAADR(STIND))
        SYNFLG(STIND)=1
        SYNLIN(STIND) = 0
        GOTO 400
C
C SYMBOL FOUND
C
    300  IF (PASS.EQ.2.OR.ICODE.EQ.1) GOTO 400
        IF (SYNFLG(STIND).NE.1) GO TO 310
        SYNFLG(STIND)=2
        CALL IACLR(SYNAADR(STIND))
        I=IADDR(SYNAADR(STIND),IADDR,SYNAADR(STIND))
        SYNLIN(STIND) = NOCARD
        GOTO 400
    310  SYNFLG(STIND)=SYNFLG(STIND).OR.8
        400  RETURN
        END
    SUBROUTINE CNVHEX(INDEX)
C
C CONVERTS 4 BITS TO HEX ASCII AND INSERTS INTO 'PL' AT 'INDEX'
C
C INPUT: WORD = VALUE
C INDEX= WHERE TO INSERT IN PL
C

```

(Continued on next page)

68000 Cross Assembler (XASM)

(Listing continued, text begins on page 12)

Listing One

```

C OUTPUT:
C      WORD = WORD/16
C
C      BYTE PL(132),DIG
C      INTEGER WORD
C      COMMON /CNVT/ WORD,PL
C      CALL GETBIT(WORD,DIG)
C      PL(INDEX)=DIG
C      RETURN
C      END
C      SUBROUTINE INSDAT(IPL,IDI6)
C
C      CONVERTS BINARY DATA TO HEX ASCII AND INSERTS INTO 'PL'
C
C      INPUT: IPL = INDEX TO INSERT INTO PL
C      IDI6= NUMBER OF DIGITS TO CONVERT AND INSERT
C      WORD= VALUE TO CONVERT (IN COMMON - NOT REFERENCED HERE)
C
C      I=IDI6
C      J=IPL+I-1
C      CALL CRWEX(J)
C      I=I-1
C      IF(I.LE.0) RETURN
C      GO TO 5
C      END
C
C      SUBROUTINE INX(15Z,IDA,IPPOS)
C
C      PRINT A 4 OR 8 DIGIT HEX VALUE
C      C NUMBER OBTAINED STARTING AT 'WORD'
C      C AND PUT INTO PRINT BUFFER 'PL' STARTING IN COL 1
C
C      IMPLICIT INTEGER (A-Z)
C      COMMON /CNVT/ WORD,PL
C      COMMON /LST/ LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERCNT
C      BYTE PL(132),NAME(8)
C      DIMENSION IDA(3)
C      PL(1)=32
C      IF(15Z.EQ.2) GO TO 15
C      WORD=IDA(1)
C      CALL INSDAT(IPPOS,4)
C      RETURN
C      WORD=IDA(2)
C      CALL INSDAT(IPPOS,4)
C      WORD=IDA(3)
C      CALL INSDAT(IPPOS,4)
C      CALL INSDAT(IPPOS,4)
C      RETURN
C      END

```

```

SUBROUTINE NEWPAG
IMPLICIT INTEGER (A-Z)
C PUTS OUT HEADERS AT TOP OF EACH PAGE
C
C      INTEGER PASS
C      BYTE NAME(8),FF
C      COMMON /LST/ LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERCNT
C      FF='14'
C      NOPAGE=NOPAGE+1
C      NOLINE = 57
C      IF(NOPAGE.EQ.1) FF = 0
C      WRITE(LUNIT,10)FF,NAME,NOPAGE
C      FORMAT(' ',1A1,8A1,T28,'68000 CROSS-ASSEMBLER V1.0
C      +',T83,'PAGE ',I3,/)
C      RETURN
C      END
C      SUBROUTINE PASCHK
C      IMPLICIT INTEGER (A-Z)
C
C      CHECKS TO SEE IF A PAGE HAS BEEN FILLED
C
C      BYTE NAME(8)
C      COMMON /LST/ LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERCNT
C      IF(NOLINE.EQ.0) CALL NEWPAG
C      RETURN
C      END
C      SUBROUTINE ERROR(IERR)
C      IMPLICIT INTEGER(A-Z)
C
C      AND PRINTS ERROR MESSAGE DURING PASS 2
C
C      COMMON /LST/ LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERCNT
C      COMMON /SRC/ LNELEN,ISERR,NOCARD,SRCLNE
C      COMMON /PSE/ OPPTR,MODPTR,OPMPTR,LABEL,CNTPTR
C      +,PRFLG,SCNPT,OPCLEN,OPNPT2,INODE
C      COMMON /OBJOUT/ OBJBUF,OBJJBC,LELG,RELG,DBFLG
C      DIMENSION OBJBUF(40)
C      COMMON /SYNT/STIND,SYNADR,PC,NOSTH,NEWPC,SYNFLG
C      INTEGER44 PC,NEWPC,SYNADR(512)
C      LOGICAL*1 SYNFLG(513),ERRPTR(80),NAME(8),SRCLNE(81)
C      LOGICAL*1 LABEL(8)
C
C      ERRORS ARE IGNORED DURING THE FIRST PASS
C
C      IF(PASS.EQ.1) RETURN
C
C      PRFLG = 3
C
C      WE NEED AT LEAST THREE LINES TO PRINT AN BAD LINE
C
C      IF(NOLINE.LE.2) NOLINE = 0
C      CALL PASCHK
C
C      IF THIS IS NOT THE FIRST ERROR THEN DON'T PRINT THE LINE
C
C      IF (NEFLG.EQ.1) GO TO 15
C      WRITE(LUNIT,10) NOCARD,(SRCLNE(I),I=1,LNELEN-1)

```

```

C
C      LSWRDS = 0B.JMC
C      IF(OBJJBC.GT.5) LSWRDS=5
C
C      CHECK IF WE HAVE TO GO TO NEXT PAGE
C
C      CALL PASCHK
C
C
C      IF(CNTPTR.NE.1)GO TO 80
C      OPPTR=1
C      GO TO 220
C      GO TO 200,200,200,410,500,600,200,200,200,400),PRFLG+1
C      CALL INX(2,PC,7)
C
C
C      IF(LSWRDS.EQ.0) GO TO 212
C      DO 210,I=1,LSWRDS
C      CALL INX(1,OBJBUF(I),11+(581))
C
C
C      IF(LABEL(1).EQ.0) GO TO 220
C      DO 215,I=1,8
C      PL(1+40)=LABEL(I)
C      J=0
C      DO 230 I=OPPTR,LNELEN
C      PL(J+50)=SRCLNE(I)
C      IF(SRCLNE(1).EQ.*40) GO TO 240
C      J=J+1
C      GO TO 1000
C      III=0
C      DO 250 II=1+1,LNELEN
C      IF (II.EQ.CNTPTR) III = 25
C      PL(57+III)=SRCLNE(II)
C      III = III + 1
C      IF ((III + 57).GT.132) GO TO 255
C      GO TO 1000
C      PL(132) = 0
C      GO TO 1000
C
C      PRFLG = 3 (NEW PAGE)
C
C      CALL NEWPAG
C      RETURN
C
C      GO TO 205
C
C      GO TO 220
C
C      CALL INX(2,OBJBUF(2),16)
C      GO TO 212
C
C      DO 1001 I=48,132

```


10	SUBROUTINE PST	1001	IF (PL(1).EQ.0) GOTO 1002
15	C SORT AND PRINT SYMBOL TABLE	1002	WRITE (LUNIT,1110) NOCARD,(PL(11),I1=6,I-1)
20	C	1110	FORMAT(' ',14,132A1)
30	C	1120	DO 1120 I1 = 1,I
	INTEGER PASS,STIND,SYNLIN(512)		PL(I1) = '40
	INTEGER4 PC,NEUPC,SYNAUR(512)		NOLINE = NOLINE - 1
	BYTE NAME(8),SYNSYN(8,512),SYNFB(513),PL(132)		RETURN
	COMMON /LST/LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERRC		END
	COMMON /SYN/SYNSYN,SYNLIN		SUBROUTINE BLDOBJ
	COMMON /SYNT/STIND,SYNAUR,PC,NOSYN,NEUPC,SYNFB		IMPLICIT INTEGER (A-Z)
	COMMON /CMT/ WORD,PL		
	IF (NOSYN.EQ.0) RETURN		BUILD OBJ FILE
			COMMON /FNAM / FILNAM,OBJFLG
	START OUT WITH CLEAN BUFFER		COMMON /OBJOUT/ OBJBUF,OBJJAC,LF6,RF6,DBFLG
50	DO 50 I = 1,132		COMMON /CMT/ WORD,PL
	PL(I) = '40		COMMON /SYNT / STIND,SYNAUR,PC,NOSYN,NEUPC,SYNFB
			COMMON /HEXFLG/ ENDFLG,HEXMC,HEXPC,OLDFC
	C GOTO TOP OF PAGE		DIMENSION OBJBUF(40),HEXBUF(8)
	CALL NEUPAG		INTEGER4 PC,NEUPC,SYNAUR(512),OLDFC,NEUVAL,HEXPC
	GENERATE THE SYMBOL LIST A LINE AT A TIME		LOGICAL #1 SYNFB(513),PL(132),FILNAM(12)
170	DO 300 I = 1,NOSYN,5		CHECK IF OBJ FILE IS TO BE GENERATED
	DO 210 IDX=0,4		IF (OBJFLG.EQ.0) RETURN
	IF (I+IDX,6T,NOSYN) GOTO 290		CHECK FOR THE END OF ASSEMBLY FLAG
	PL(IPT+1)=7,2		IF IT IS SET, WRITE OUT THE BALANCE OF THE OBJ BUFFER
	DO 170 IPT=1,7,2		IF (ENDFLG.EQ.0) GOTO 10
	PL(IPT+(IDXX24)+1) = SYNSYN(IPT,(I+IDXX))		IF (HEXMC.NE.0) CALL WRTOBJ(HEXPC,HEXMC,HEXBUF)
	PL(IPT+(IDXX24)) = SYNSYN(IPT+1,(I+IDXX))		RETURN
	CALL INX(2,SYNAUR(I+IDXX),(IDXX24)+12)		CHECK THE CURRENT VALUE OF THE PC WITH THAT OF THE ONE SAVED
	IFTRP = SYNFB(I+IDXX)		IF THE TWO ARE NOT EQUAL, THEN WRITE OUT THE BALANCE OF THE
	IF ((IFTRP.AND.16).NE.16) GOTO 180		OBJ BUFFER AND START AT THE NEW PC VAL
180	PL((IDXX24)+19) = 'E'		CALL DBLSL(PC,PC1,PC2)
	PL((IDXX24)+20) = 'Q'		CALL DBLSL(OLDFC,OLDFC1,OLDFC2)
	IF ((IFTRP.AND.8).NE.8) GOTO 190		IF (PC1.NE.OLDFC1) GOTO 50
	PL((IDXX24)+19) = 'N'		IF (PC2.EQ.OLDFC2) GOTO 75
190	PL((IDXX24)+20) = 'U'		IF (HEXMC.NE.0) CALL WRTOBJ(HEXPC,HEXMC,HEXBUF)
	IF ((IFTRP.AND.1).NE.1) GOTO 200		CALL JMOV(PC,HEXPC)
	PL((IDXX24)+19) = 'U'		CALL JMOV(PC,OLDFC)
	PL((IDXX24)+20) = 'N'		EXTRACT OBJECT WORDS FROM OBJECT BUFFER AND
200	IF ((IFTRP.AND.'31').NE.0) GOTO 210		PUT THEM INTO AN INTERNAL BUFFER, IF THE
	PL((IDXX24)+19) = ' '		INTERNAL BUFFER IS FULL, THEN OUTPUT THE BUFFER.
	PL((IDXX24)+20) = ' '		
210	CONTINUE		
290	WRITE (LUNIT,400) (PL(N),N=1,IDXX24)		
	NOLINE = NOLINE -1		
300	CALL PASCHK		
400	CONTINUE		
	FORMAT(' ',132A1)		
410	WRITE (LUNIT,410) NOSYN,IERRC		
	FORMAT('/',',13',SYMBOLS,'13,' ERRORS DETECTED')		
	IF (LUNIT.EQ.5) RETURN		
	WRITE (5,410) NOSYN,IERRC		
	RETURN		
	END		

(Continued on page 18, column 2)

(Continued on page 18, column 3)

(Continued on next page)

68000 Cross Assembler (XASM)

(Listing continued, text begins on page 12)

Listing One

```

75 I = 1
76 HEXMC = HEXMC + 1
   HEXBUF(HEXMC) = OBJBUF(1)
   IF (HEXMC.NE.8) GOTO 99
C.... OBJECT BUFFER IS FULL, OUTPUT IT TO OBJ FILE
C
C CALL WRTOBJ(HEXPC,HEXMC,HEXBUF)
C
C CALCULATE NEW STARTING PC FOR HEX BUFFER
C
C N = JICVT(142,NEWVAL)
C N = JADD(PC,NEWVAL,HEXPC)
C I = I + 1
C IF (I.LE.OBJMC) GOTO 76
C
C CALCULATE WHAT THE NEW PC SHOULD BE BY ADDING
C THE OBJECT WORD COUNT TO THE CURRENT PC
C
C I = JADD(OLDFPC,NEWPC,OLDPC)
C RETURN
C END
C SUBROUTINE WRTOBJ(HEXPC,HEXMC,HEXBUF)
C IMPLICIT INTEGER(A-Z)
C
C OUTPUT THE CONTENTS OF THE OBJECT BUFFER
C
C HEXPC = STARTING PC FOR BUFFER
C HEXMC = NUMBER OF WORDS USED IN BUFFER
C HEXBUF = 8 WORD OBJECT BUFFER
C
C COMMON /CMT/ WORD,PL
C LOGICAL X1 PL(132)
C INTEGER X4 HEXPC
C DIMENSION HEXBUF(8)
C DO 10, I = 1,80
C PL(I) = '40'
C CALL IXH(2,HEXPC,I)
C PLIDX = 10
C DO 20, I=1,HEXMC
C CALL IXH(1,HEXBUF(I),PLIDX+(8*(I-1)))
C WRITE (2,100) (PL(I),I=3,10+(8*HEXMC))
C FORMAT(' ',80A1)
C HEXMC = 0
C DO 900, I = 1,80
C PL(I) = '40'
C RETURN
C END
C SUBROUTINE PROCESS

```

```

C COMMON /OPND / OPNFG,OPNMC,OPNMD
C
C COMMON /LST / LUNIT,PASS,NAME,NOPAGE,NOLINE,NEFLG,IERCNT
C
C COMMON /OBJOUT/ OBJBUF,OBJMC,FLG,RFLG,BRFLG
C
C COMMON /SRC / LNELEN,ISERR,MOCARD,SRCLE
C
C COMMON /SYMT / STIND,SYMAOR,PC,MOSYN,NEWPC,SYNFG
C
C COMMON /PRSE / OPNTR,MOPNTR,OPNTR,LABEL,CNTPTR
C +,PRFLG,SCAMPT,OPCLEN,OPNPT2,INODE
C
C COMMON /OPCPTS/ OPTYP,OPSKEL,OPSK2,OPIDX
C
C INTEGER X4 PC,NEWPC,SYMAOR(512),SYNVAL,TMPVAL,J2
C LOGICAL X1 SRCLE(81),LABEL(8),NAME(8),SYNFG(513)
C DIMENSION OBJBUF(40),OPNMD(3)
C
C SET UP FLAGS THAT CHANGE EACH TIME THRU
C....
C CALL IACLR(NEWPC)
C J2 = 2
C OP1EA = 0
C OP2EA = 0
C OP1DA = 0
C OP2DA = 0
C OPNMC = 0
C
C DECODE OPNMC
C....
C CALL DECODE
C IF (OPTYP.NE.0) GOTO 10
C CALL ERROR(400)
C RETURN
C
C SKIP IF NO OPERANDS
C....
C IF (OPNTR.EQ.0) GOTO 20
C DECODE FIRST OPERAND
C....
C OP1EA=OPNTR
C CALL EATYP(OP1EA,OP1DA)
C IF (OPNPT2.EQ.0) GOTO 20
C DECODE SECOND OPERAND
C....
C OP2EA=OPNPT2
C CALL EATYP(OP2EA,OP2DA)
C
C CHECK FOR OPERANDS
C....
C IF (OP1EA.EQ.10.OR.OP2EA.EQ.10) GOTO 8500
C IF (OPTYP.EQ.1.OR.OPTYP.EQ.2) GOTO 90
C IF (OPNTR.NE.0) GOTO 90
C CALL ERROR(401)

```

```

IF (INODE.NE.3) GOTO 125
I=JICVT(4,NEWPC)
I=JNIL(NEWPC,OPNMD(2),NEWPC)
GOTO 128
122 I=JMOV(OPNMD(2),NEWPC)
GOTO 128
125 I=JICVT(2,NEWPC)
I=JNIL(NEWPC,OPNMD(2),NEWPC)
I=JMOV(PC,OBJBUF(2))
I=JMOV(PC,SYNVAL)
GOTO 7005
C
C DRG
C
C IF (LABEL(1).EQ.0) GOTO 132
C CALL ERROR(402)
C RETURN
C
C RFLG=1
C PRFLG=7
C IF (OPNTR.NE.0) GOTO 134
C CALL IACLR(NEWPC)
C CALL IACLR(PC)
C RETURN
C CALL PROCOP(OPNTR)
C IF (OPNMC.EQ.7) GOTO 135
C CALL IACLR(PC)
C I=JADD(NEWPC,OPNMD(2),NEWPC)
C RETURN
C CALL ERROR(403)
C RETURN
C
C END <STARTING ADDR>
C
C ISERR=1
C IF (LABEL(1).NE.0) GOTO 131
C PRFLG=5
C RETURN
C
C EQU
C
C IF (LABEL(1).EQ.0) GOTO 131
C PRFLG=6
C IF (OPNTR.EQ.0) GOTO 8500
C CALL PROCOP(OPNTR)
C IF (OPNMC.EQ.7) RETURN
C CALL SYMTBL(2,OPNMD(2),LABEL)
C IF ((SYNFG(STIND).AND.'10).EQ.'10) CALL ERROR(409)
C I=JMOV(OPNMD(2),SYMAOR(STIND))
C SYNFG(STIND)=SYNFG(STIND).OR.16
C I=JMOV(SYMAOR(STIND),OBJBUF(2))
C RETURN
C
C RORG
C
C IF (LABEL(1).NE.0) GOTO 131
C RFLG=0
C GOTO 133

```


PROCESSES SOURCE LINE AFTER IT HAS BEEN PARSED BY PARSE

68000 Cross Assembler (XASM)

(Listing continued, text begins on page 12)

Listing One

```

C      IF(OP1EA, EQ, 7, OR, OP1EA, EQ, 9) GOTO 350
C      OP1EA = 1 THRU 5
C      IF((OP1EA, GE, 1), AND, (OP1EA, LE, 5)) GOTO 305
C      PROCESS FIRST OPM HERE IF COMPLEX
C      CALL PROCOP(OPMPTR)
C      CHECK FOR EA TYPES 7-9
C      IF(OP2EA, GT, 6) GOTO 340
C      CHECK FOR FIRST OPERAND IMMEDIATE MODE ADDRESSING
C      IF (OP1EA, NE, 6) GOTO 304
C      SKIP MOVQ IF FWD REF SYMBOL
C      IF(OPNFG, EQ, 1) GOTO 304
C      IF(INODE, NE, 3) GOTO 304
C      IF(OPNWD(3), EQ, 0) GOTO 301
C      IF(OPNWD(3), EQ, -1) GOTO 301
C      GOTO 304
C      CHECK IF VAL WITHIN RANGE FOR MOVED (+/- 128)
C      ALSO CHECK IF DESTINATION IS A DATA REGISTER
C      I=ICXVAL(OPNWD(2))
C      IF ((I, EQ, 0), AND, (OP2EA, EQ, 1)) GOTO 330
C      ADD IN OPCODE SIZE BITS
C      OBJBUF(1)=OBJBUF(1), OR, *30000
C      IF(INODE, EQ, 1) OBJBUF(1)=(OBJBUF(1)), AND, *17777
C      IF(INODE, EQ, 3) OBJBUF(1)=(OBJBUF(1)), AND, *27777
C      MOVE IN NUMBERS FOR 1ST AND 2ND EXT WORDS
C      OBJMC = OBJMC+OPNMC
C      OBJBUF(2) = OPNWD(2)
C      OBJBUF(1) = OPNWD(1)
C      IF(OPNMC, EQ, 2) OBJBUF(2) = OPNWD(3)
C      IF(OPNMC, EQ, 2) OBJBUF(3) = OPNWD(2)
C      GOTO 310
C      PROCESS EA TYPES 0-5 FOR FIRST OPM
C      GOTO 7000
C      HANDLE STUFF FOR EA'S 0 AND 6
C      OBJBUF(1)=OBJBUF(1), OR, OPNWD(1)
C      OBJBUF(2)=OPNWD(2)
C      IF(OPNMC, EQ, 2) OBJBUF(2)=OPNWD(3)
C      IF(OPNMC, EQ, 2) OBJBUF(3)=OPNWD(2)
C      OBJMC=OBJMC+OPNMC
C      GOTO 7000
C      GENERATE MOVE SR<EA> - USP, AM
C      IF (OP1EA, EQ, 9) GOTO 355
C      IF (OP2EA, EQ, 2) GOTO 8500
C      IF (OP2EA, EQ, 0) GOTO 353
C      OBJBUF(1) = *40300, OR, OP2DA, OR, ((OP2EA-1)*10)
C      GOTO 7000
C      CALL PROCOP(OPMPTR)
C      OBJBUF(2)=OPNWD(2)
C      IF(OPNMC, EQ, 2) OBJBUF(2) = OPNWD(3)
C      IF(OPNMC, EQ, 2) OBJBUF(3) = OPNWD(2)
C      OBJMC = OBJMC + OPNMC
C      OBJBUF(1) = *43000, OR, OPNWD(1)
C      GOTO 7000
C      OBJBUF(1) = *47150, OR, OP2DA
C      GOTO 7000
C      ++++++
C      PROCESS CMP INSTRUCTION
C      <EA>, DM <EA>, AM DATA<EA> (AY)+, (AX)+
C      ++++++
C      IF(OP1EA, EQ, 6), AND, (OP2EA, NE, 2)) GOTO 460
C      IF((OP1EA, EQ, 5), AND, (OP2EA, EQ, 5)) GOTO 480
C      IF((OP2EA, EQ, 1), OR, (OP2EA, EQ, 2)) GOTO 410
C      GOTO 8500
C      PROCESS <EA>, DM <EA>, AM
C      IF(OP2EA, EQ, 2, AND, INODE, EQ, 1) GOTO 8500
C      IF(OP2EA, ME, 2) GOTO 411
C      IF(INODE, EQ, 3) OPSKEL = OPSKEL, OR, *500
C      IF(INODE, ME, 3) OPSKEL = OPSKEL, OR, *200
C      IF((OP1EA, EQ, 0), OR, (OP1EA, EQ, 6)) GOTO 415
C      PROCESS FOR REG OPNS
C      OBJBUF(1)=OPSKEL, OR, (OP2DA)*1000, OR, ((OP1EA-1)*10), OR, OP1DA
C      GOTO 6000
C      PROCESS FOR COMPLEX 1ST OPNS
C      GOTO 7000
C      GENERATE DM<EA>
C      OPSKEL = OPSKEL, OR, (OP1DA)*1000
C      IF(OP2EA, EQ, 0) GOTO 511
C      OBJBUF(1) = OPSKEL, OR, ((OP2EA-1)*10), OR, OP2DA
C      GOTO 6000
C      CALL PROCOP(OPMPTR)
C      OBJBUF(2) = OPNWD(2)
C      IF(OPNMC, EQ, 2) OBJBUF(2) = OPNWD(3)
C      IF(OPNMC, EQ, 2) OBJBUF(3) = OPNWD(2)
C      OBJMC = OBJMC+OPNMC
C      OBJBUF(1) = OPSKEL, OR, OPNWD(1)
C      GOTO 6000
C      GENERATE <EA>, DM
C      OPSKEL = OPSKEL, OR, (OP2DA)*1000
C      IF(OP1EA, EQ, 0) GOTO 522
C      OBJBUF(1) = OPSKEL, OR, ((OP1EA-1)*10), OR, OP1DA
C      GOTO 6000
C      CALL PROCOP(OPMPTR)
C      GOTO 514
C      GENERATE <EA>, AM
C      IF (INODE, EQ, 1) GOTO 8500
C      IF (INODE, EQ, 3) OPSKEL = OPSKEL, OR, *500
C      IF ((INODE, EQ, 2), OR, (INODE, EQ, 0)) OPSKEL = OPSKEL, OR, *200
C      OPSKEL = OPSKEL, OR, (OP2DA)*1000
C      IF((OP1EA, EQ, 0), OR, (OP1EA, EQ, 6)) GOTO 522
C      OBJBUF(1) = OPSKEL, OR, ((OP1EA-1)*10), OR, OP1DA
C      GOTO 6000
C      GENERATE xxxI
C      IF(OP2EA, GT, 6) GOTO 8500
C      EVALUATE IMMEDIATE EXPRESSION
C      CALL PROCOP(OPMPTR)
C      TRY GENERATING SHORT FORM OF INSTRUCTION
C      AFTER CHECKING TO SEE IF OPERAND WAS FWD REF
C      IF(OPNFG, EQ, 1) GOTO 536
C      IF(OPNWD(2), GE, 1, AND, OPNWD(2), LE, 8) GOTO 550
C      GENERATE EXTENSION WORDS
C      LENGTH OF OPERAND DEPENDS ON THE INODE OF INSTRUCTION
C      OBJBUF(2) = OPNWD(2)
C      IF(OPNMC, EQ, 2) OBJBUF(2) = OPNWD(3)
C      IF(OPNMC, EQ, 2) OBJBUF(3) = OPNWD(2)
C      OBJMC = OBJMC + OPNMC
C      GOTO 537

```



```

C
305  C.....  OB:JBUF(1)=((OP1EA-1)*10).OR.(OP1DA)
C
C.....  CHK FOR SIMPLE SECOND OPERANDS
C
310  C.....  IF(OP2EA.EQ.0) GOTO 315
C
C.....  CHK FOR SR,CCR,USP
C
C.....  IF(OP2EA.GT.6) GOTO 340
C
C.....  GOTO 320
C
C.....  CALCULATE COMPLEX SECOND OPN
C
315  C.....  CALL PROCOP(OPNPT2)
C
C.....  IF(OPNMC.EQ.2) OB:JBUF(OB:JMC+2)=OPNARD(2)
C.....  IF(OPNMC.EQ.2) OB:JBUF(OB:JMC+1)=OPNARD(3)
C.....  OB:JMC=OB:JMC+OPNMC
C.....  I=(OPNARD(1).AND.7)*10
C.....  J=(OPNARD(1).AND.*70)/8
C.....  OB:JBUF(1)=OB:JBUF(1).OR.((I+J)*100).OR.*30000
C.....  GOTO 325
C
C.....  PROCESS EA TYPES 0-5 FOR SECOND OPN
C
320  C.....  OB:JBUF(1)=OB:JBUF(1)+((OP2EA-1).OR.(OP2DA*10))*100).OR.*30000
C
C.....  ADD IN SIZE BITS
C
325  C.....  IF(INDOE.EQ.1)OB:JBUF(1)=OB:JBUF(1).AND.*17777
C.....  IF(INDOE.EQ.3)OB:JBUF(1)=OB:JBUF(1).AND.*27777
C.....  GOTO 7000
C
C.....  GEN MOVED ALSO CLR SIZE BITS IF SET
C
330  C.....  OB:JBUF(1) = 0
C.....  OB:JBUF(1) = (OPNARD(2).AND.*377).OR.*70000).OR.(OP2DA*1000)
C.....  GOTO 7000
C
C.....  GENERATE MOVE <EA>,SR - <EA>,CCR - AM,USP
C
340  C.....  IF(OP2EA.EQ.7) OB:JBUF(1)=*43300
C.....  IF(OP2EA.EQ.8) OB:JBUF(1)=*42300
C.....  IF(OP2EA.EQ.9) GOTO 342
C.....  OB:JBUF(1) = *47140).OR.OP1DA
C.....  GOTO 7000
C
C.....  GET NON-REG EA'S IF 0 OR 6
C
342  C.....  IF(OP1EA.EQ.0).OR.(OP1EA.EQ.6) GOTO 349
C
C.....  ELSE JUST ADD OR IN THE EA AND REG
C
C.....  OB:JBUF(1)=OB:JBUF(1).OR.OP1DA).OR.((OP1EA-1)*10)
C
C.....  CALL PROCOP(OPNPT2)
C.....  IF(BEST THRU REG EVAL IT HERE)
C
538  C.....  IF(OP2EA.EQ.0) GOTO 540
C.....  OB:JBUF(1)=OPSK2).OR.((OP2EA-1)*10).OR.OP2DA
C.....  GOTO 6000
C
C.....  EVAL NON-REG DEST
C
540  C.....  CALL PROCOP(OPNPT2)
C.....  OB:JMC = OB:JMC + OPNMC
C.....  OB:JBUF(1) = OPSK2).OR.OPNARD(1)
C.....  IF(OPNMC.EQ.1) OB:JBUF(OB:JMC) = OPNARD(2)
C.....  IF(OPNMC.EQ.2) OB:JBUF(OB:JMC+1) = OPNARD(3)
C.....  IF(OPNMC.EQ.2) OB:JBUF(OB:JMC) = OPNARD(2)
C.....  GOTO 6000
C
C.....  GENERATE xxx0
C
550  C.....  IF(OPNARD(2).EQ.8) OPNARD(2) = 0
C.....  IF(OPSK2.EQ.*2000) OPSK2 = *50400
C.....  IF(OPSK2.EQ.*3000) OPSK2 = *50000
C.....  OPSK2 = OPSK2).OR.(OPNARD(2)*1000)
C.....  GOTO 538
C
C.....  PROCESS AND/OR INSTRUCTIONS
C.....  <EA>,DN DN,<EA> DATA,<EA>
C
C.....  IF(OP1EA.EQ.6) GOTO 610
C.....  IF(OP2EA.EQ.1) GOTO 620
C.....  PROCESS <EA>,DN
C
C.....  OPSKEL=OPSKEL+(OP2DA*1000)
C.....  IF(OP1EA.EQ.0) GOTO 605
C.....  OB:JBUF(1)=OPSKEL).OR.OP1DA).OR.((OP1EA-1)*10)
C.....  GOTO 6000
C
C.....  CALL PROCOP(OPNPT2)
C.....  OB:JBUF(1)=OPSKEL).OR.OPNARD(1)
C.....  OB:JBUF(2)=OPNARD(2)
C.....  IF(OPNMC.EQ.2) OB:JBUF(2)=OPNARD(3)
C.....  IF(OPNMC.EQ.2) OB:JBUF(3)=OPNARD(2)
C.....  OB:JMC=OB:JMC+OPNMC
C.....  GOTO 6000
C
C.....  PROCESS DATA,<EA>
C
610  C.....  OPSKEL = OPSK2
C.....  IF(OP2EA.EQ.6) GOTO 8500
C.....  CALL PROCOP(OPNPT2)

```

(Continued on next page)

(Continued on page 22, column 3)

(Continued on page 22, column 2)

68000 Cross Assembler (XASM)

(Listing continued, text begins on page 12)

Listing One

```

C      OBJBUF(2)=OPWARD(2)
C      IF(OPNMC.EQ.2) OBJBUF(2)=OPWARD(3)
C      IF(OPNMC.EQ.2) OBJBUF(3)=OPWARD(2)
C      OBJMC=OBJMC+OPNMC
C
C      NOW THAT WE HAVE IMMEDIATE DATA GET <EA>
C
C      IF(OP2EA.EQ.0.AND.OP1EA.EQ.1) GOTO 6000
C      IF(OP2EA.EQ.0) GOTO 615
C
C      CHECK FOR DATA+SR OR DATA+CCR
C
C      IF(OP2EA.LT.7) GOTO 612
C      IF(OP2EA.GT.8) GOTO 8500
C      IF((IMODE.EQ.1).AND.(OP2EA.EQ.8)) GOTO 611
C      IF((IMODE.EQ.1).OR.(IMODE.EQ.3)) GOTO 8500
C
611    OBJBUF(1) = OPSKEL+OR.*74
C      GOTO 6000
612    OBJBUF(1) = OPSKEL+OR.((OP2EA-1)*10).OR.OP2DA
C      GOTO 6000
C
C      EVALUATE <EA> FOR COMPLEX ADDR
C
C      CALL PROCOP(OPNPT2)
C      OBJBUF(OBJMC+1)=OPWARD(2)
C      IF(OPNMC.EQ.2) OBJBUF(OBJMC+1)=OPWARD(3)
C      IF(OPNMC.EQ.2) OBJBUF(OBJMC+2)=OPWARD(2)
C      OBJMC=OBJMC+OPNMC
C      OBJBUF(1)=OBJBUF(1).OR.OPSKEL
C      GOTO 6000
C
C      EVALUATE DM<EA>
C
C      OPSKEL=OPSKEL+(OP1DA*1000).OR.*400
C      IF(OP2EA.EQ.0) GOTO 615
C      OBJBUF(1) = OPSKEL+OR.OP2DA.OR.((OP2EA-1)*10)
C      GOTO 6000
C
C      ++++++
C
C      PROCESS EOR INSTRUCTION
C      DM<EA> DATA<EA>
C
C      ++++++
C      IF(OP1EA.EQ.6) GOTO 610
C      IF(OP1EA.NE.1) GOTO 8500
C      IF(OP2EA.EQ.0) GOTO 620
C      OBJBUF(1)=OPSKEL+((OP1EA-1)*1000)+OP2DA+((OP1EA-1)*10)

```

(Continued on column 2)

```

C      GOTO 6000
C      ++++++
C      PROCESS ROTATES AND SHIFTS
C      DX+DY DATA+DY <EA>
C
C      ++++++
C      IF(OP1EA.EQ.1.AND.OP2EA.EQ.1) GOTO 810
C      IF(OP1EA.EQ.6.AND.OP2EA.EQ.1) GOTO 820
C      IF(OP1EA.EQ.0.AND.OP2EA.EQ.1) GOTO 820
C
C      PROCESS <EA>
C
C      IF(OP1EA.EQ.0) GOTO 801
C      IF(OP1EA.LT.3.OR.OP1EA.GT.5) GOTO 8500
C      OBJBUF(1)=OPSKEL+((OP1EA-1)*10)+OP1DA
C      GOTO 7000
C
C      CALL PROCOP(OPNPT2)
C      OBJBUF(1)=OPSKEL+OPWARD(1)
C      OBJBUF(2) = OPWARD(2)
C      IF(OPNMC.EQ.2) OBJBUF(2) = OPWARD(3)
C      IF(OPNMC.EQ.2) OBJBUF(3) = OPWARD(2)
C      OBJMC = OBJMC + OPNMC
C      GOTO 7000
C
C      OBJBUF(1) = OPSKEL+OR.*40.OR.(OP1DA*1000).OR.OP2DA
C      GOTO 6000
C
C      CALL PROCOP(OPNPT2)
C      IF(OPNPT2.LT.1.OR.OPNPT2.GT.8) GOTO 8500
C      IF(OPNPT2.EQ.8) OPWARD(2)=0
C      OBJBUF(1)=OPSKEL+OPWARD(2)*1000+OP2DA
C      GOTO 6000
C      ++++++
C
C      PROCESS BRANCH INSTRUCTIONS
C      <LABEL>
C
C      ++++++
C      IF(OPNPT2.EQ.0) GOTO 8500
C      IF(OP1EA.NE.0) GOTO 8500
C      BRFLG = 1
C
C      GENERATE BRANCH ADDRESS
C
C      CALL PROCOP(OPNPT2)
C
C      CHK FOR FORCED SHORT ADDR MODE
C
C      IF(IMODE.EQ.4) GOTO 910
C
C      CHECK FOR FWD REF SYMBOL OR REF BEFORE DEFINITION

```

(Continued on column 3)

```

C      IF(OPNPLG.EQ.1) GOTO 905
C
C      CHECK FOR SHORT BRANCH
C
C      I = ICVVAL(OPWARD(2))
C      IF((1.EQ.0).AND.(OPNPT2.NE.*177600)) GOTO 910
C      IF(IMODE.EQ.4) CALL ERROR(404)
C
C      ELSE GENERATE TWO WORD BRANCH
C
C      OBJBUF(1) = OPSKEL
C      OBJBUF(2) = OPWARD(2)
C      OBJMC = 2
C      GOTO 920
C
C      GENERATE SHORT BRANCH
C
C      OBJMC = 1
C      OPSKEL=OPSKEL+OPWARD(2).AND.*377)
C      OBJBUF(1) = OPSKEL
C      BRFLG = 0
C      GOTO 7000
C      ++++++
C
C      PROCESS BIT MODIFICATION INSTRUCTIONS
C      DM<EA> DATA<EA>
C
C      ++++++
C      IF(OP1EA.EQ.1.OR.OP1EA.EQ.6) GOTO 1010
C      GOTO 8500
C      IF(OP1EA.EQ.6) GOTO 1020
C      IF(OP2EA.EQ.0) GOTO 1015
C
C      SIMPLE EA'S
C
C      OBJBUF(1) = OPSKEL+OR.(OP1DA*1000).OR.OP2DA
C      OBJBUF(1) = OBJBUF(1).OR.((OP2EA-1)*10)
C      GOTO 7000
C
C      CALL PROCOP(OPNPT2)
C      OBJBUF(2) = OPWARD(2)
C      IF(OPNMC.EQ.2) OBJBUF(2) = OPWARD(3)
C      IF(OPNMC.EQ.2) OBJBUF(3) = OPWARD(2)
C      OBJMC = OBJMC + OPNMC
C      OBJBUF(1) = OPSKEL+OR.OPWARD(1).OR.(OP1DA*1000)
C      GOTO 7000
C
C      CALL PROCOP(OPNPT2)
C      IF(OPNPT2.NE.0) GOTO 8500
C      OBJBUF(2)=OPWARD(2)

```

(Continued next month.)

The Portable Pidgin

Z80 Macro-Assembly Implementation

When in doubt, write in a machine-independent language. Anyone who has seen a cherished program lie fallow for lack of portability will recognize the wisdom of this maxim. Yet machine-independent languages are normally high-level affairs ill-suited for supporting the compact code and rapid execution times required of many systems programs. Whence my excitement upon learning of William Gale's *Pidgin*, a low-level, structured, portable programming language (DDJ #57). I suspect that Dr. Gale's compiler generator *Meta4* (DDJ #58) is only the first of many powerful systems programs that will be written in the flexible format of *Pidgin*.

by Herbert Gintis

*Dr. Herbert Gintis, Dept. of Economics,
Harvard University, Cambridge, MA
02138.*

Pidgin is portable because it was designed to be supported by a portable compiler. Indeed, Dr. Gale has provided us with *Tincmp*, a *Pidgin* compiler written in *Pidgin* itself, yet organized to facilitate its relatively easy implementation on any machine. But only relatively! What I shall suggest here is that *Tincmp* can be implemented with little difficulty on any machine for which a reasonably powerful macro-assembler exists. The example presented here uses a Z80 microprocessor running under CP/M, and the macroprocessor is *Venus*, a structured editor-assembler-linker-symbolic debugger which I've developed over the past year.

The strategy for implementing *Tincmp* is straightforward: generate an assembly-language version of the compiler by translating each and every *Pidgin* statement into a macro.

The result of this exercise is a new *Tincmp* source file which I have called

Tinsource, shown in Listing 2, and a source file of macros called *Tinmacro*, shown in Listing 1. I shall discuss these two source files in turn.

Since *Venus* uses standard assembly-language pseudo-ops (e.g., CP/M's ASM-COM), *Tinsource* should be easy to understand, subject to the following notes. First, *Venus* uses the pseudo-op INCBG <filename.filetype> to instruct the assembler to include the specified file at the beginning of the source code before assembly. In this case, *Tinmacro* will be so included. Second, a string of one to five alphanumeric characters beginning with a letter, occurring where *Venus* expects a mnemonic opcode or pseudo-op, and followed by an exclamation point (!), is treated as a macro call.

The meaning of the macros is either the same as that of the corresponding *Pidgin* statement, is indicated in the remarks, or can be inferred by comparing *Tinsource* with *Tincmp*. I have attempted to use obvious mnemonics where possible. Thus "B" stands for "byte", "I" for "int", "E" for "set equal to", "O" for "or", "A" for "and", "EE" for "is equal to", "LE" for "is less than or equal to", and so forth.

Since *Venus* is a structured assembler, I in fact implemented the control statements WHILE-ON-ENDWHILE and IF-ELSE-ENDIF directly. However, since structured facilities are rare in assemblers, I have rewritten *Tinsource* by treating these control statements as program labels. Thus a WHILE becomes the string "WH" followed by a number, and ENDWHILE becomes "NDWH" followed by a number. The ON in the WHILE-ENDWHILE construction is translated into the macro IFNOT, which moves beyond the ENDWHILE if the condition tested is false. The IF-ELSE-ENDIF is similarly treated, using the common beginning string IF followed by a number. The CHOOSE ON-CASE construction is directly implemented as a macro.

Tinmacro, given in Listing 1 is also by and large self-explanatory. The macro-processor of *Venus* is fairly standard. The name of the macro is written in the label column followed by the pseudo-op MACRO, followed in turn by a list of the prototype variables separated by commas. A prototype variable is preceded by the ampersand (&). The period (.) is a non-printed string terminator.

NEW IBM PERSONAL COMPUTER SOFTWARE

C88 SOFTWARE DEVELOPMENT PACKAGE — \$250

The C88 Software Development Package includes the C88 Compiler, the L88 Linkage Editor, and extensive library of useful routines and functions. Some of the features of the package are:

- Compiles an extensive subset of the C language
- Extremely fast compilation
- Allows linkage of separately compiled modules
- Library includes numerous useful routines.

DMS DATA MANAGEMENT SYSTEM — \$150

The DMS Data Management System is a package which allows simple and quick development of user data entry Forms, and allows retrieval, modification, and reporting of Data Bases using these Forms. Some of the features of DMS are:

- User defined screen formats
- Efficient B-Tree key index searching
- Human engineered user interface
- Data files readable by BASIC, PASCAL and C88 programs.

PCTEXT TEXT FORMAT PROCESSOR — \$100

The PCTEXT text processor allows easy production of well formulated documents, while allowing use of any text editor or word processor to prepare the text. Some of the features of PCTEXT are:

- Indentation and centering
- Line spacing and margin control
- Headings, footings and pagination control
- Allows imbedding of one document in another

To order direct, enclose your check or money order. New York State residents add applicable local sales tax. Shipped prepaid within the United States.

Also Available

- "Window" full screen text editor and "Datacomm" data communications program

Write Department DD for detailed information on any of the above.

i.a.i.

INTELLECT ASSOCIATES INC.
MICROCOMPUTER SPECIALISTS

P.O. BOX 365
HOLBROOK, NY 11741

Circle no. 226 on reader service card.

Thus if the macroprocessor encounters the line

```
LD HL, (I.&DEST)
```

and the current value of &DEST is GG, the line will be expanded as

```
LD HL, (IGG)
```

Tinmacro is organized according to the principle that each byte variable in the object code generated (which is in this case an assembly source file) is preceded by the string BB, and each integer variable is preceded by the single letter I. By comparing the listing of Tinmacro with the 8080A macros suggested by Andrew Bender (DDJ #65 - March 1982), the reader will notice that I have streamlined some of the coding, using the fact that the Z80 accumulator can be loaded without altering the flag conditions.

To implement Tincmp for your machine, alter Tinmacro and Tinsource to meet the conventions of your assembler.

If you do not have a pseudo-op like INCBG, then Tinmacro will have to be merged directly with Tinsource, an easy task on a text editor. If you do not have a macroprocessor, you are probably out of luck, since the full source code of Tinsource with all macros expanded would probably run some 4000 lines.

When Tinsource assembles without errors, then you must implement the Tincmp macros, following the example of Bender's, perhaps with the streamlining suggested in Listing 1. Note that Bender's implementation of the structured control statements uses multiple ORG's, which are disliked by some relocating assemblers. Venus does not mind such multiple ORG's, but there is an easy alternative strategy which I have successfully tested anyway.

The problem arises with the CHOOSE ON \$\$ and CASE \$\$ statements in Pidgin. Unlike the C language, Pidgin does not allow a program to "fall through" to

the next CASE upon the execution of one CASE, but rather demands exiting the control structure. Thus in generating the assembly source code for the CASE \$\$ statement, the first line must be a jump over any remaining CASE \$\$ statements. However, this jump must be suppressed in the *first* such statement after the CHOOSE ON \$\$. The strategy for handling this is to have the CHOOSE ON \$\$ place a comment symbol (;) on the top of the stack, and have each CASE \$\$ immediately place the top of the stack in the destination file, and replace it with a blank (), which will be ignored by the assembler. The relevant Tincmp macros will then be:

```
:CHOOSE ON $$;
LD A, (BB↑P1C↑P2C);
LD HL, ERASER;
LD (HL), A↑U0S↑L@; S↑U1S;
:CASE $$;
↑!0P↑!9P↑!8P;
↑P9CJP NDCH↑P0N;
CS↑P8N LD A, (BB↑P1C↑P2C);
CP (HL);
JP NZ, CS↑U1S↑S8N↑L@ S↑P0S;
```

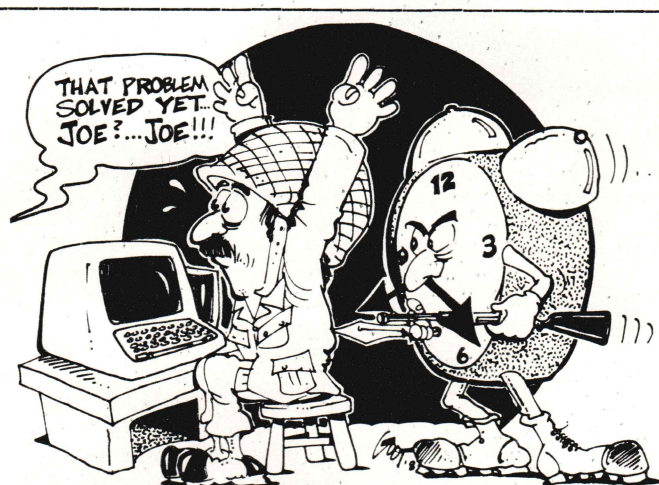
The final step in implementing Tincmp involves developing the system utilities, including the algebraic routines ICOMP, ISUB, CDEHL, IMUL, and IDIV, as well as the CP/M input-output routines START, BREAD, BWRITE, PRT, IOPEN, MSG, CONOUT, and BCLOSE. The requirements for the algebraic routines are well-described in Bender's excellent article. CONOUT is essentially CP/M's Direct Consol I/O (Function 6), but preserving all registers. PRT uses CONOUT to output the string pointed to by the HL register; the string must terminate with a carriage return (0dh in ASCII). MSG is the same as PRT, except that precisely 9 characters are transmitted. The I/O routines must be written using the system utilities associated with your assembler. Their functions, will be obvious from the places in Listing 1 where they are used, after a careful reading of the discussion of these routines in Bender's article.

DDJ

(Listing begins at right)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 514



LOSING TRACK OF CHANGES? "NOW! COMPARE files on CPM-80!"

- Compare Program Versions
- Fast, NO File Restrictions
- Side by Side or Vertical Listings of Differences
- Output to File or Printer
- Compare Documents
- Extensive User Options: Ignore Comments, Blank lines and more
- Create Documents with Change Bars
- Sensible Command Line defaults

COMPARE is a superb software tool with excellent features for the serious professional programmer with no time to waste. Document mode with Change Bars designed especially with Writers in mind.

Call or Write for Information Only \$95

SOLUTION TECHNOLOGY, INC.

Suite 218 • 1499 Palmetto Park Rd. • Boca Raton, FL 33432 • (305) 368-6228

*CPM-80 is a trademark of Digital Research Inc. *Check or COD, Florida residents add 5% sales tax.

Circle no. 299 on reader service card.

Tinmacro (Text begins on page 25)

Listing One

```

(A:TINMACRO)11 11,f
1 * *****
2 * TINMACRO *****
3 *
4 * *****
5 *
6 * Listing 1
7 *
8 * *****
9 *
10 * Macros for the implementation of TINSOURCE
11 *
12 IIEI MACRO &A,&B,&C ;I$=I$($)=I$
13 LD HL,(I.&B)
14 LD DE,I.&A
15 ADD HL,HL
16 ADD HL,DE
17 EX DE,HL
18 LD HL,(I.&C)
19 LD A,L
20 LD (DE),A
21 INC DE
22 LD A,H
23 LD (DE),A
24 MEND
25 IINC MACRO &A ;I$++
26 LD HL,(I.&A)
27 INC HL
28 LD (I.&A),HL
29 MEND
30 BIEB MACRO &A,&B,&C ;I$($)=I$
31 LD HL,(I.&B)
32 LD DE,BE.&A
33 ADD HL,DE
34 LD A,(BE.&C)
35 LD (HL),A
36 MEND
37 BINC MACRO &A ;I$++
38 LD HL,BE.&A
39 INC (HL)
40 MEND
41 BBEB MACRO &A,&B,&C ;I$($)=I$
42 LD HL,(BE.&B)
43 LD DE,BE.&A
44 ADD HL,DE
45 LD A,(BE.&C)
46 LD (HL),A
47 MEND
48 REEB MACRO &A,&B,&C ;I$=$($)=I$
49 LD HL,(BE.&C)
50 LD H,0
51 LD DE,BE.&B
52 ADD HL,DE
53 LD A,(HL)
54 LD (BE.&A),A
55 MEND
56 IEIPI MACRO &A,&B,&C ;I$=I$+I$
57 LD HL,(I.&B)

```

```

59 LD DE,(I.&C)
60 ADD HL,DE
61 LD (I.&A),HL
62 MEND
63 IEI MACRO &A,&B,&C ;I$=I$($)=I$
64 LD HL,(I.&C)
65 LD DE,I.&B
66 ADD HL,HL
67 ADD HL,DE
68 LD E,(HL)
69 INC HL
70 LD D,(HL)
71 EX DE,HL
72 LD (I.&A),HL
73 MEND
74 BEBI MACRO &A,&B,&C ;I$=$($)=I$
75 LD HL,(I.&C)
76 LD DE,BE.&B
77 ADD HL,DE
78 LD A,(HL)
79 LD (BE.&A),A
80 MEND
81 BEB MACRO &A,&B ;I$=$($)=I$
82 LD A,(BE.&B)
83 LD (BE.&A),A
84 MEND
85 IEI MACRO &A,&B ;I$=I$($)=I$
86 LD HL,(I.&B)
87 LD (I.&A),HL
88 MEND
89 BECON MACRO &A,&B ;I$=$($)=I$
90 LD A,BE
91 LD (BE.&A),A
92 MEND
93 IECON MACRO &A,&B ;I$=$($)=I$
94 LD HL,BE
95 LD (I.&A),HL
96 MEND
97 BEBEB MACRO &A,&B,&C ;I$=$($)=I$
98 LD HL,BE.&C
99 LD A,(BE.&B)
100 ADD A,(HL)
101 LD (BE.&A),A
102 MEND
103 BEBMB MACRO &A,&B,&C ;I$=$($)=I$
104 LD HL,BE.&C
105 LD A,(BE.&B)
106 SUB (HL)
107 LD (BE.&A),A
108 MEND
109 BEBAB MACRO &A,&B,&C ;I$=$($)=I$
110 LD HL,BE.&C
111 LD A,(BE.&B)
112 AND (HL)
113 LD (BE.&A),A
114 MEND
115 BEBOB MACRO &A,&B,&C ;I$=$($)=I$
116 LD HL,BE.&C
117 LD A,(BE.&B)
118 OR (HL)
119 LD (BE.&A),A
120 MEND
121 IEMI MACRO &A,&B ;I$=I$-I$
122 LD HL,(I.&B)
123 ICOMP
124 LD (I.&A),HL

```

(Continued on next page)

(Continued on column 2)

Tinmacro (Listing continued, text begins on page 25)

Listing One (Continued)

```

125 ICOMP      EXTRN
126           MEND
127 IEIMI       MACRO
128           LD      &A,&B,&C ; $$$=I$$-I$$
129           HL,(I.&C)
130           EX      DE,HL
131           LD      HL,(I.&B)
132           CALL   ISUB
133           LD      (I.&A),HL
134           EXTRN
135           MEND
136           MACRO
137           LD      &BB,&IAA,&IBB ; $$$=I$$!=I$$
138           DE,I.&IBB
139           HL,I.&IAA
140           INTCK  (BB.&BB),A
141           EXTRN
142           MEND
143           MACRO
144           LD      &BB,&IAA,&IBB ; $$$=I$$(<=I$$
145           HL,(I.&IBB)
146           EX      DE,HL
147           LD      HL,(I.&IAA)
148           CDEHL
149           SETAC
150           EXTRN
151           LD      (BB.&BB),A
152           MACRO &BB ; BYTE $$
153           DS      1
154           BB.&BB
155           REL
156           MEND
157           MACRO &IAA ; INT I$$
158           RAM
159           DS      2
160           I.&IAA
161           REL
162           MACRO &B1,&NNN ; BYTE $$(I$$) OR BYTE $$(I$$)
163           RAM
164           DS      &NNN
165           REL
166           MEND
167           MACRO &IAA,&NNN ; INT I$$(I$$)
168           RAM
169           DS      2 * &NNN
170           REL
171           MEND
172           MACRO &ROUTINE
173           CALL   SUB.&ROUTINE
174           MEND
175           MACRO &N
176           LD      HL,$+6
177           CALL   PR
178           JP      0000
179           DB      'STOP ON ',&N+0',:D
180           MEND
181           MACRO &BB,&IAA,&IBB ; $$$=I$$(<I$$
182           LD      HL,(I.&IBB)
183           EX      DE,HL
184           LD      HL,(I.&IAA)
185           CDEHL
186           EXTRN
187           RLA

```

```

254 CP (HL)
255 RLA
256 AND 1
257 LD (BB.&B1),A
258 MEND
259 IEIB        MACRO
260           LD      &IAA,&IBB,&B1 ; $$$=I$$($$)
261           HL,(BB.&B1)
262           H,0
263           LD      DE,I.&IBB
264           ADD     HL,HL
265           ADD     HL,DE
266           LD      DE,I.&IAA
267           A,(HL)
268           LD      (DE),A
269           INC     HL
270           INC     DE
271           LD      A,(HL)
272           LD      (DE),A
273           MEND
274           MACRO &B1,&IAA ; $$$=I$$
275           LD      A,(I.&IAA)
276           LD      (BB.&B1),A
277           MEND
278           MACRO &B1,&B2,&B3 ; $$$=I$$(<=I$$
279           HL,BB.&B3
280           A,(BB.&B2)
281           CP      (HL)
282           SETAC  (BB.&B1),A
283           EXTRN
284           MEND
285           MACRO &B1 ; $$$--
286           LD      HL,BB.&B1
287           LD      (HL)
288           MEND
289           MACRO &IAA,&IBB,&ICC ; $$$=I$$*I$$
290           LD      HL,(I.&ICC)
291           DE,HL
292           LD      HL,(I.&IBB)
293           CALL   IMUL
294           LD      (I.&IAA),HL
295           EXTRN
296           MEND
297           MACRO &IAA,&IBB,&ICC ; $$$=I$$/I$$
298           LD      HL,(I.&IBB)
299           DE,HL
300           LD      HL,(I.&ICC)
301           CALL   IDIV
302           LD      (I.&IAA),HL
303           EXTRN
304           MEND
305           MACRO &IAA,&B1,&ICC ; $$$($$)=I$$
306           LD      HL,(BB.&B1)
307           H,0
308           ADD     HL,HL
309           LD      DE,I.&IAA
310           ADD     HL,DE
311           LD      DE,I.&ICC
312           LD      A,(DE)
313           LD      (HL),A
314           INC     DE
315           INC     HL
316           LD      A,(DE)
317           LD      (HL),A
318           MEND
319           ITBB  &IAA,&B1,&B2 ; UNPACK(I$$,$$,I$$)
320           LD      HL,(I.&IAA)
321           LD      A,L
322           LD      (BB.&B1),A

```



```

188 AND
189 LD (BB.&BB),A
190 MEND
191 MACRO
192 LD &CC,&F1 ;READ $$ FROM $$
193 LD HL,BB.&CC
194 DE,BB.&F1
195 BREAD
196 CALL
197 EXTRN
198 MEND
199 MACRO
200 LD &CC,&F1 ;WRITE $$ INTO $$
201 A,(BB.&CC)
202 DE,BB.&F1
203 BWRITE
204 CALL
205 LD A,(ER)
206 OR A
207 NZ,PRT
208 CALL
209 EXTRN
210 MEND
211 MACRO
212 LD &F1,&RW,&INM ;OPEN $$ FOR $$ AT I$$
213 DE,BB.&F1
214 A,(BB.&RW)
215 HL,(I.&INM)
216 IOPEN
217 LD A,(ER)
218 OR A
219 NZ,PRT
220 CALL
221 EXTRN
222 MEND
223 MACRO
224 LD &STRING ;PRINT 9 CHARS
225 HL,$+5
226 MSG
227 JR
228 DB
229 &STRING
230 MS.&XX,FF
231 MEND
232 MACRO
233 LD &B1,&B2,&B3 ;$$=$$=$$
234 HL,BB.&B3
235 A,(BB.&B2)
236 (HL)
237 A,0
238 Z,1
239 INC A
240 (BB.&B1),A
241 MEND
242 MACRO
243 LD &B1,&B2,&B3 ;$$=$$=$$
244 HL,BB.&B3
245 A,(BB.&B2)
246 (HL)
247 A,1
248 JR
249 Z,1
250 XOR A
251 (BB.&B1),A
252 MEND
253 MACRO
254 LD &IAA,&B1 ;I$$=$$
255 A,(BB.&B1)
256 L,A
257 H,0
258 (I.&IAA),HL
259 MEND
260 MACRO
261 LD &B1,&B2,&B3 ;$$=$$($$)
262 A,(BB.&B2)
263 HL,BB.&B3

```

```

323 LD
324 A,H
325 (BB.&B2),A
326 MEND
327 MACRO
328 LD &B1,&IAA,&IBB ;$$=I$$=I$$
329 HL,(I.&IAA)
330 DE,(I.&IBB)
331 CDEHL
332 LD A,1
333 Z,1
334 XOR A
335 (BB.&B1),A
336 MEND
337 MACRO
338 LD &NN ;GOTO $$
339 LOC.&NN
340 JP
341 MEND
342 MACRO
343 LD &B1 ;WRITE $$
344 A,(BB.&B1)
345 CONOUT
346 MEND
347 MACRO
348 LD &B1 ;CLOSE $$
349 HL,BB.&B1
350 BCLOSE
351 LD A,(ER)
352 OR A
353 NZ,PRT
354 CALL
355 EXTRN
356 MEND
357 MACRO
358 LD &B1 ;CHOOSE ON $$
359 A,(BB.&B1)
360 MEND
361 MACRO
362 LD &B1,&DEST ;GOTO DEST IF B1 FALSE
363 A,(BB.&B1)
364 OR A
365 Z,&DEST
366 MEND
367 MACRO
368 LD &B1,&DEST ;GOTO DEST IF [A] != [B1]
369 HL,BB.&B1
370 (HL)
371 NZ,&DEST
372 JP
373 MEND
374 (A:TINMACRO).ff

```

Tinsource Listing Two

```

(A:TINMACRO).p
(A:TINCOMP).11,1,f
1 * ***** TINSOURCE *****
2 *
3 *
4 * *****
5 *
6 * LISTING 2
7 *
8 * *****
9 *
10 * The TINCMP Compiler in Assembly Language
11 *
12 *
13 UNLST ; suppress printer listing
14 INCBG TINMACRO ; append TINMACRO here
15 LIST ; resume printer listing

```

(Continued on top of page 28)

(Continued on next page)

Tinsource (Listing continued, text begins on page 25)

Listing Two (Continued)

```

15 ER XMAC
16 EXTRN
17 EQU ER
18 BYTE! AA
19 BYTE! BB
20 BYTE! DD
21 BYTE! EE
22 BYTE! BF,080
23 BYTE! BL
24 BYTE! BP
25 BYTE! C0
26 BYTE! C1
27 BYTE! C2
28 BYTE! C3
29 BYTE! C4
30 BYTE! C8
31 BYTE! C9
32 BYTE! CC
33 BYTE! CX
34 *
35 BYTE! DG
36 BYTE! DS,10
37 BYTE! EF
38 BYTE! F1,120
39 BYTE! F2,128
40 BYTE! HA
41 BYTE! HF
42 BYTE! LE
43 BYTE! LF
44 BYTE! LS,3000
45 BYTE! MF
46 BYTE! ML
47 BYTE! MM
48 *
49 BYTE! ND
50 BYTE! NL
51 BYTE! O1
52 BYTE! O2
53 BYTE! O3
54 BYTE! OA
55 BYTE! OB
56 BYTE! OC
57 BYTE! OD
58 BYTE! OE
59 BYTE! OG
60 BYTE! OH
61 BYTE! OL
62 BYTE! OM
63 BYTE! ON
64 *
65 BYTE! OP
66 BYTE! OR
67 BYTE! OS
68 *
69 BYTE! OT
70 BYTE! PP
71 BYTE! RB
72 BYTE! RC
73 BYTE! SF
74 BYTE! SP
75 BYTE! TR
76 BYTE! UG
77 BYTE! UN
78 BYTE! UD

; suppress the printing of macro expansion
; all byte variables begin with BB
;WORK
;WORK
;WORK
;WORK
;EXPANSION BUFFER
;BLANK
;PTR INTO BF
;0
;1
;2
;3
;40
;80
;9
;INPUT CHAR
;10
;DIGIT FROM PARAMETER DEF
;DIGIT STACK FOR SUB SD
;END OF FILE MARK
;INPUT BUFFER
;OUTPUT BUFFER
;'A'
;'F'
;END OF LIST
;LINE FEED CHAR
;LIST OF MACRO DEFS
;MACRO REPLACEMENT OPERATOR FLAG
;MACRO LENGTH
;MIN. MACRO LENGTH

;# OF DIGITS IN SD OUTPUT
;NEW LINE
;FETCH CODE
;INDEX CODE
;DISPOSE CODE
;'+' OPERATOR
;'-' OPERATOR
;'C' CHAR. DISPOSE OPERATOR
;'V' DIGIT CONV. FETCH
;ESC CHAR
;IGNORE CHAR
;'H' HEX CONV. FETCH
;'L' LITERAL FETCH
;'BYTE!' MULT DISPOSE
;'N' NUMERIC LITERAL FETCH

;'P' PARAMETER FETCH OR DISPOSE
;'-' REDUCE (SUBTRACT) DISPOSE
;'S' STACK FETCH OR DISPOSE

;TRACE FLAG
;PTR INTO IPR
;BEGIN DEF. FLAG
;END OF LINE (COMMENT) FLAG
;SUBS PARAMETER FLAG
;STACK PTR
;TRUE IF NO TRACE
;USE IGNORE: TRUE UNLESS QG='X'
;FLAG FOR NOT SUPPRESSING NEW LINESON OUTPUT
;USE OPERATIONS: TRUE UNLESS MF='X'

```



```

79  BYTE! UT
80  BYTE! ZR
81  INT! I00
82  INT! I01
83  INT! I09
84  INT! I10
85  INT! I16
86  INT! IAA
87  INT! IBB
88  INT! IBC
89  INT! IDP
90  INT! IED
91  INT! IJI
92  INT! IJJ
93  INT! ILM
94  INT! ILP, I000
95  INT! IMP
96  INT! INM
97  INT! IPR, I0
98  INT! ISS, I0
99  INT! ITU
100 INT! IUU
101 INT! IXX
102 INT! IYY
103 * BEGINMAIN(AC, IAV)
104 CALL START
105 EXTRN
106 BECON! NL, :D
107 BECON! LF, :A
108 GOSUB! CR
109 MS! 'Copyright'
110 MS! ' (C) 1982'
111 MS! ' Herbert '
112 MS! ' Gintis '
113 GOSUB! CR
114 MS! 'Z80 TINCMP'
115 MS! 'P Compiler'
116 MS! 'r CPM Ver'
117 MS! 'sion 1.1 '
118 GOSUB! CR
119 GOSUB! IN
120 GOSUB! RM
121 LOC00
122 WH7
123 GOSUB! GI
124 BEEB! AA, ER, C0
125 IFNOT! AG, NDWH7
126 REEB! BB, CC, NL
127 REEB! DD, CC, LF
128 REEB! AA, CC, OG
129 REEB! EE, BB, DD
130 REEB! BB, CC, BL
131 REEB! AA, BB, AA
132 REEB! AA, EE, AA
133 IFNOT! AA, NDWH8
134 GOSUB! GC
135 JR WH8
136 NDWH8
137 IF31
138
139 REEB! BP, C1
140 REEB! BF, C0, CC
141 GOSUB! GC
142 REEB! AA, CC, NL
143 IFNOT! AA, IF32
144 GOSUB! GC
145 REEB! AA, CC, LF
146 IFNOT! AA, IF33

```

```

215 IFNOT! AA, NDWH13
216 IFNOT! ML, IF38
217 WRITE! CC
218 BEB! AA, CC
219 GOSUB! WA
220 BINC! BP
221 JP WH13
222 NDWH13
223 IFNOT! ML, IF39
224 GOSUB! CR
225 IFNOT! UN, IF40
226 BEB! AA, NL
227 GOSUB! WA
228 BEB! AA, LF
229 GOSUB! WA
230 IF40
231 JP WH7
232 NDWH7
233 LOC88
234 MS! 'TINCMP CO'
235 MS! 'MPILATION'
236 GOSUB! CR
237 CLOSE! F1
238 CLOSE! F2
239 JP :I00
240 * Initialize
241 *
242 *
243 SUBIN
244 IECON! ILM, 8920 ; ILM=+08920
245 IECON! I00, 0
246 IECON! I01, 1
247 IECON! I10, I0
248 IECON! I09, 9
249 BECON! C0, 0 ; CO=+000
250 BECON! C1, 1
251 BECON! C2, 2
252 BECON! C3, 3
253 BECON! EF, :IA
254 BECON! C4, 40
255 BECON! C8, 80
256 IECON! I16, 16
257 BECON! SP, 0
258 BECON! C9, :9'
259 BECON! BL, : '
260 BECON! HF, :F'
261 BECON! HA, :A'
262 BECON! CX, I0
263 IEI! IBC, I01 ; IBC=I01
264 BECON! TR, :R'
265 * CLOSE F1
266 * ASSOCIATE FCB 1 WITH IBC
267 OPEN! F1, TR, IBC ; OPEN F1 FOR TR AT IBC
268 BECON! TR, :W' ; TR=:W'
269 IINC! IBC ; IBC++
270 * CLOSE F2
271 * ASSOCIATE FCB 2 WITH IBC
272 OPEN! F2, TR, IBC
273 GOSUB! GI
274 BEB! AA, CC
275 BECON! OT, :T'
276 BECON! UT, :0
277 BECON! BB, :X'
278 BEB! UN, AA, BB
279 GOSUB! GI
280 BEB! RB, CC
281 GOSUB! GI

```

;READ AA FROM F1;X' SUPPRESSES NEW LINE OUTPUT

;UN=AA'=BB;TRUE IF NOT SUPPRESS
;BEGIN DEF FLAG
;COMMENT & EOL FLAG

(Continued on top of page 30)

(Continued on next page)

32

Listing Two (Continued)

```

262 BEB! RC, CC
263 GOSUB! GI
264 BEB! SF, CC
265 GOSUB! GI
266 BEB! MF, CC
267 *
268 * Set UD true iff expansion flag is not 'X'
269 *
270 BECON! BE, 'X'
271 REEB! AA, MF, BB
272 IFNOT! AA, IF1
273 JR IF2
274 BEB! UD, C1
275 IF1
276 IF2
277 BECON! BE, 'X'
278 REEB! AA, MF, BB
279 IFNOT! AA, IF1
280 JR IF2
281 BEB! UD, C1
282 IF1
283 IF2
284 BEB! RC, CC
285 GOSUB! GI
286 BEB! SF, CC
287 GOSUB! GI
288 BEB! MF, CC
289 *
290 BECON! BE, 'X'
291 REEB! AA, MF, BB
292 IFNOT! AA, IF1
293 JR IF2
294 BEB! UD, C1
295 IF1
296 IF2
297 BECON! BE, 'X'
298 REEB! AA, MF, BB
299 IFNOT! AA, IF1
300 JR IF2
301 BEB! UD, C1
302 IF1
303 IF2
304 BEB! RC, CC
305 GOSUB! GI
306 BEB! SF, CC
307 GOSUB! GI
308 BEB! MF, CC
309 *
310 BECON! BE, 'X'
311 REEB! AA, MF, BB
312 IFNOT! AA, IF1
313 JR IF2
314 BEB! UD, C1
315 IF1
316 IF2
317 IF4
318 BECON! BE, 'X'
319 REEB! AA, MF, BB
320 IFNOT! AA, IF1
321 JR IF2
322 BEB! UD, C1
323 IF1
324 IF2
325 *
326 * Read Macros
327 *
328 SUBRM
329 IEI!
330 IEB! INM, C0
331 BECON! MM, 127
332 WH1
333 GOSUB! GI
334 REEB! AA, ER, C0
335 IFNOT! AA, NDWH1
336 CHON! CC
337 CASE! OE, CS1
338 GOSUB! GI
339 JP LOC77
340 CASE! RB, CS2
341 IEI! ILP, INM, III
342 INC! INM
343 BECON! ML, 0
344 JP NDCH1
345 CASE! NL, CS2A
346 JP NDCH1
347 *
348 * Read Macros
349 *
350 SUBRM
351 IEI!
352 IEB! INM, C0
353 BECON! MM, 127
354 WH1
355 GOSUB! GI
356 REEB! AA, ER, C0
357 IFNOT! AA, NDWH1
358 CHON! CC
359 CASE! OE, CS1
360 GOSUB! GI
361 JP LOC77
362 CASE! RB, CS2
363 IEI! ILP, INM, III
364 INC! INM
365 BECON! ML, 0
366 JP NDCH1
367 CASE! NL, CS2A
368 JP NDCH1
369 *
370 * Read Macros
371 *
372 SUBRM
373 IEI!
374 IEB! INM, C0
375 BECON! MM, 127
376 WH1
377 GOSUB! GI
378 REEB! AA, ER, C0
379 IFNOT! AA, NDWH1
380 CHON! CC
381 CASE! OE, CS1
382 GOSUB! GI
383 JP LOC77
384 CASE! RB, CS2
385 IEI! ILP, INM, III
386 INC! INM
387 BECON! ML, 0
388 JP NDCH1
389 CASE! NL, CS2A
390 JP NDCH1
391 *
392 * Read Macros
393 *
394 SUBRM
395 IEI!
396 IEB! INM, C0
397 BECON! MM, 127
398 WH1
399 GOSUB! GI
400 REEB! AA, ER, C0
401 IFNOT! AA, NDWH1
402 CHON! CC
403 CASE! OE, CS1
404 GOSUB! GI
405 JP LOC77
406 CASE! RB, CS2
407 IEI! ILP, INM, III
408 INC! INM
409 BECON! ML, 0
410 JP NDCH1
411 CASE! NL, CS2A
412 JP NDCH1
413 *
414 * Read Macros
415 *
416 SUBRM
417 IEI!
418 IEB! INM, C0
419 BECON! MM, 127
420 WH1
421 GOSUB! GI
422 REEB! AA, ER, C0
423 IFNOT! AA, NDWH1
424 CHON! CC
425 CASE! OE, CS1
426 GOSUB! GI
427 JP LOC77
428 CASE! RB, CS2
429 IEI! ILP, INM, III
430 INC! INM
431 BECON! ML, 0
432 JP NDCH1
433 CASE! NL, CS2A
434 JP NDCH1
435 *
436 * Read Macros
437 *
438 SUBRM
439 IEI!
440 IEB! INM, C0
441 BECON! MM, 127
442 WH1
443 GOSUB! GI
444 REEB! AA, ER, C0
445 IFNOT! AA, NDWH1
446 CHON! CC
447 CASE! OE, CS1
448 GOSUB! GI
449 JP LOC77
450 CASE! RB, CS2
451 IEI! ILP, INM, III
452 INC! INM
453 BECON! ML, 0
454 JP NDCH1
455 CASE! NL, CS2A
456 JP NDCH1
457 *
458 * Read Macros
459 *
460 SUBRM
461 IEI!
462 IEB! INM, C0
463 BECON! MM, 127
464 WH1
465 GOSUB! GI
466 REEB! AA, ER, C0
467 IFNOT! AA, NDWH1
468 CHON! CC
469 CASE! OE, CS1
470 GOSUB! GI
471 JP LOC77
472 CASE! RB, CS2
473 IEI! ILP, INM, III
474 INC! INM
475 BECON! ML, 0
476 JP NDCH1
477 CASE! NL, CS2A
478 JP NDCH1
479 *
480 * Read Macros
481 *
482 SUBRM
483 IEI!
484 IEB! INM, C0
485 BECON! MM, 127
486 WH1
487 GOSUB! GI
488 REEB! AA, ER, C0
489 IFNOT! AA, NDWH1
490 CHON! CC
491 CASE! OE, CS1
492 GOSUB! GI
493 JP LOC77
494 CASE! RB, CS2
495 IEI! ILP, INM, III
496 INC! INM
497 BECON! ML, 0
498 JP NDCH1
499 CASE! NL, CS2A
500 JP NDCH1
501 *
502 * Read Macros
503 *
504 SUBRM
505 IEI!
506 IEB! INM, C0
507 BECON! MM, 127
508 WH1
509 GOSUB! GI
510 REEB! AA, ER, C0
511 IFNOT! AA, NDWH1
512 CHON! CC
513 CASE! OE, CS1
514 GOSUB! GI
515 JP LOC77
516 CASE! RB, CS2
517 IEI! ILP, INM, III
518 INC! INM
519 BECON! ML, 0
520 JP NDCH1
521 CASE! NL, CS2A
522 JP NDCH1
523 *
524 * Read Macros
525 *
526 SUBRM
527 IEI!
528 IEB! INM, C0
529 BECON! MM, 127
530 WH1
531 GOSUB! GI
532 REEB! AA, ER, C0
533 IFNOT! AA, NDWH1
534 CHON! CC
535 CASE! OE, CS1
536 GOSUB! GI
537 JP LOC77
538 CASE! RB, CS2
539 IEI! ILP, INM, III
540 INC! INM
541 BECON! ML, 0
542 JP NDCH1
543 CASE! NL, CS2A
544 JP NDCH1
545 *
546 * Read Macros
547 *
548 SUBRM
549 IEI!
550 IEB! INM, C0
551 BECON! MM, 127
552 WH1
553 GOSUB! GI
554 REEB! AA, ER, C0
555 IFNOT! AA, NDWH1
556 CHON! CC
557 CASE! OE, CS1
558 GOSUB! GI
559 JP LOC77
560 CASE! RB, CS2
561 IEI! ILP, INM, III
562 INC! INM
563 BECON! ML, 0
564 JP NDCH1
565 CASE! NL, CS2A
566 JP NDCH1
567 *
568 * Read Macros
569 *
570 SUBRM
571 IEI!
572 IEB! INM, C0
573 BECON! MM, 127
574 WH1
575 GOSUB! GI
576 REEB! AA, ER, C0
577 IFNOT! AA, NDWH1
578 CHON! CC
579 CASE! OE, CS1
580 GOSUB! GI
581 JP LOC77
582 CASE! RB, CS2
583 IEI! ILP, INM, III
584 INC! INM
585 BECON! ML, 0
586 JP NDCH1
587 CASE! NL, CS2A
588 JP NDCH1
589 *
590 * Read Macros
591 *
592 SUBRM
593 IEI!
594 IEB! INM, C0
595 BECON! MM, 127
596 WH1
597 GOSUB! GI
598 REEB! AA, ER, C0
599 IFNOT! AA, NDWH1
600 CHON! CC
601 CASE! OE, CS1
602 GOSUB! GI
603 JP LOC77
604 CASE! RB, CS2
605 IEI! ILP, INM, III
606 INC! INM
607 BECON! ML, 0
608 JP NDCH1
609 CASE! NL, CS2A
610 JP NDCH1
611 *
612 * Read Macros
613 *
614 SUBRM
615 IEI!
616 IEB! INM, C0
617 BECON! MM, 127
618 WH1
619 GOSUB! GI
620 REEB! AA, ER, C0
621 IFNOT! AA, NDWH1
622 CHON! CC
623 CASE! OE, CS1
624 GOSUB! GI
625 JP LOC77
626 CASE! RB, CS2
627 IEI! ILP, INM, III
628 INC! INM
629 BECON! ML, 0
630 JP NDCH1
631 CASE! NL, CS2A
632 JP NDCH1
633 *
634 * Read Macros
635 *
636 SUBRM
637 IEI!
638 IEB! INM, C0
639 BECON! MM, 127
640 WH1
641 GOSUB! GI
642 REEB! AA, ER, C0
643 IFNOT! AA, NDWH1
644 CHON! CC
645 CASE! OE, CS1
646 GOSUB! GI
647 JP LOC77
648 CASE! RB, CS2
649 IEI! ILP, INM, III
650 INC! INM
651 BECON! ML, 0
652 JP NDCH1
653 CASE! NL, CS2A
654 JP NDCH1
655 *
656 * Read Macros
657 *
658 SUBRM
659 IEI!
660 IEB! INM, C0
661 BECON! MM, 127
662 WH1
663 GOSUB! GI
664 REEB! AA, ER, C0
665 IFNOT! AA, NDWH1
666 CHON! CC
667 CASE! OE, CS1
668 GOSUB! GI
669 JP LOC77
670 CASE! RB, CS2
671 IEI! ILP, INM, III
672 INC! INM
673 BECON! ML, 0
674 JP NDCH1
675 CASE! NL, CS2A
676 JP NDCH1
677 *
678 * Read Macros
679 *
680 SUBRM
681 IEI!
682 IEB! INM, C0
683 BECON! MM, 127
684 WH1
685 GOSUB! GI
686 REEB! AA, ER, C0
687 IFNOT! AA, NDWH1
688 CHON! CC
689 CASE! OE, CS1
690 GOSUB! GI
691 JP LOC77
692 CASE! RB, CS2
693 IEI! ILP, INM, III
694 INC! INM
695 BECON! ML, 0
696 JP NDCH1
697 CASE! NL, CS2A
698 JP NDCH1
699 *
700 * Read Macros
701 *
702 SUBRM
703 IEI!
704 IEB! INM, C0
705 BECON! MM, 127
706 WH1
707 GOSUB! GI
708 REEB! AA, ER, C0
709 IFNOT! AA, NDWH1
710 CHON! CC
711 CASE! OE, CS1
712 G
```

```

412 WRITE! LF
413 RET
414 *
415 * Do Macro Expansion
416 *
417 SUBDM
418
419
420 WH3
421
422
423
424
425
426 IF11
427 IF12
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442 IF15
443
444
445
446
447 CS4A
448
449
450
451
452
453 CS5
454
455
456
457
458
459
460 IF16
461
462
463 CS6
464
465
466 CS7
467
468
469
470
471
472
473
474
475
476
477 CS8
478
479
480 CS9

```

```

IMP,IJJ,IQ1 ;IMP=IJJ+IQ1
INM
IDP,ILP,INM ;IDP=ILP(INM)
ISLI! AA,IMP,IDP ;AA=IMP(IDP
IFNOT! AA,NDWH3
BEBI! AA,LS,IMP ;AA=LS(IMP)
IFNOT! UU,IF11
BEER! AA,AA,MF ;AA=AA+MF
JR IF12
BEB! AA,CQ

IFNOT! AA,IF13
INM! IMP
BEB! O1,LS,IMP ;O1=LS(IMP)
INM! IMP
BEB! AA,LS,IMP ;AA=LS(IMP)
BEB! O2,AA
GOSUB! CD
BEB! DG,AA
INM! IMP
BEB! O3,LS,IMP
IFNOT! UT,IF15
WRITE! O1
WRITE! O2
WRITE! O3

CHSON! O1 ;CHOOSE ON O1
CASE! OP,CS4A ;FETCH PARAMETER
IEIB! ITU,IPR,DG ;ITU=IPR(DG)
JP NDCH2
CASE! OD,CS5 ;CONVERT FROM DIGIT TO CHAR
IEIB! IAA,IPR,DG ;IAA=IPR(DG)
BEB! AA,IAA ;AA=IAA
JP NDCH2
IEB! ITU,AA ;ITU=AA
JP NDCH2
CASE! OB,CS6 ;POP STACK
IEIB! ITU,ISS,SP ;ITU=ISS(SP)
BLEB! AA,SP,CQ ;AA=SP=CQ
IFNOT! AA,IF16
MS! 'S STACKER'
GOSUB! CR
BEB! SP,C1

BDED! SP
JP NDCH2
CASE! OS,CS7 ;FETCH TOP STACK, NO POP
IEIB! ITU,ISS,SP ;ITU=ISS(SP)
JP NDCH2
CASE! OH,CS8 ;FETCH & WRITE HEX CONST. BYTE
BEB! AA,O2
GOSUB! CH
IEB! IAA,AA ;IAA=AA
ITMS! IAA,IAA,I16 IAA=IAA*I16
BEB! AA,O3
GOSUB! CH
IEB! IBB,AA
IEIP! ITU,IAA,IEB
BEB! O3,OC
JP NDCH2
CASE! OL,CS9 ;LITERAL BYTE FETCH
IEB! ITU,O2
JP NDCH2
CASE! ON,CS10 ;LITERAL DIGIT FETCH

```

```

;ENDSUB CR

```



```

346 CS2A CASE! LF,CS2B ; IGNORE LF
347 JP NDCH1
348 CS2B CASE! RC,CS3 ; IGNORE COMMENTS
349 BIEB! LS,III,RC ;LS(III)=RC
350 IINC! III
351 BSLB! AA,ML,MM ;AA=ML<MM
352 IFNOT! AA,IF6
353 BEB! MM,ML
354 IF6 GOSUB! G1
355 WH2 BNEB! AA,CC,LF ;AA=CC=LF
356 IFNOT! AA,NDWH2
357 JR WH2
358 NDWH2
359
360 JP NDCH1
361 CS3 CASE! OG,CS4 ;IF USING IGNORE, IGNORE
362 IFNOT! UG,IF7
363 JR IF8
364 IF7 JP LOC77
365 IF8
366 JP NDCH1
367 CS4 REM
368 LOC77
369
370 BIEB! LS,III,CC ;LS(III)=CC
371 IINC! III
372 ISLI! AA,ILM,III ;AA=ILM<III
373 IFNOT! AA,IF9
374 MS! 'MACMEMXST'
375 GOSUB! CR
376 CLOSE! F1
377 STOP! 5
378 IF9
379 NDCH1 BINC! ML
380
381 JP WH1
382 NDWH1
383
384 RNEB! AA,CC,EF ;AA=CC=EF
385 IFNOT! AA,IF10
386 MS! 'DEFN READ'
387 STOP! 2
388 IF10
389
390 CLOSE! F1
391 IECON! IRC,3
392 * ASSOCIATE FCB 3 WITH IBC
393
394 BECON! TR,'R'
395 OPEN! F1,TR,IBC
396 IEI! IED,III ;IED=III ;END OF DEFS
397 MS! 'LOADED...'
398 IEI! ITU,III
399 GOSUB! PN
400 MS! 'BYTES FO'
401 MS! 'R DEFINES'
402 GOSUB! CR
403 IIEI! ILP,INM,III ;ILP(INM)=III
404 IEI! ITU,INM
405 GOSUB! PN
406 MS! 'MACROS...'
407 IEI! ITU,MM ;ITU=MM
408 GOSUB! PN
409 MS! 'MIN LEN.'
410 GOSUB! CR
411 RET
412
413 * Do carriage return
414 *
415 SUBCR WRITE! NL

```

```

481 REB! AA,02
482 GOSUB! CD
483 IEI! ITU,AA
484 JP NDCH2
485 CS10 CASE! OT,CS11 ;TURN ON TRACE MODE
486 BECON! UT,1 ;UT=+001
487 JR NDCH2
488 REM
489 IEI! ITU,IUU
490 IINC! IUU
491 NDCH2
492 IFNOT! UT,IF17
493 IEI! III,ITU
494 GOSUB! PN
495 IEI! ITU,ISS,SP ;ITU=ISS(SP)
496 GOSUB! PN
497 IEI! ITU,SP ;ITU=SP
498 GOSUB! PN
499 IEI! ITU,III
500 GOSUB! CR
501 IF17
502 CHSON! 03 ;CHOOSE ON 03
503 CASE! OC,CS12 ;CHAR OUTPUT
504 BEI! AA,ITU
505 GOSUB! WA
506 JP NDCH3
507 CS12 CASE! OS,CS13 ;PUT ON STACK
508 BINC! SP
509 BLEB! AA,C4,SP
510 IFNOT! AA,IF18
511 MS! 'S OVERFLOW'
512 GOSUB! CR
513 BEB! SP,C4
514 IF18
515 ISS,SP,ITU
516 JP NDCH3
517 CS13 CASE! OP,CS14 PUT INTO PARAMETER LOCATION
518 IEI! IPR,DG,ITU
519 JP NDCH3
520 CS14 CASE! OA,CS15 ;ADD TO STACK
521 IFIR! IAA,ISS,SP
522 IEIPI! IAA,IAA,ITU ;IAA=IAA+ITU
523 IEI! ISS,SP,IAA
524 JP NDCH3
525 CS15 CASE! OR,CS16 ;SUBTRACT FROM STACK
526 IEI! IAA,ISS,SP
527 IEIMI! IAA,IAA,ITU ;IAA=IAA-ITU
528 IEI! ISS,SP,IAA
529 JP NDCH3
530 CS16 CASE! OM,CS17 ;MULTIPLY BY BASE AND ADD
531 IEI! IAA,ISS,SP
532 IIMS! IAA,IAA,II0 ;IAA=IAA*II0
533 IEIPI! IAA,IAA,ITU
534 IEI! ISS,SP,IAA
535 JP NDCH3
536 CS17 CASE! OH,CS18 ;OUTPUT HIGH BYTE
537 ITBB! ITU,AA,BB ;UNPACK(ITU,AA,BB)
538 GOSUB! WA
539 JR NDCH3
540 CS18 GOSUB! WN ;WRITE AS DECIMAL #
541 NDCH3
542 IF14
543 IF13 REM
544 BEI! AA,LS,IMP
545 UN,IF19
546 BNEB! BB,AA,RC ;BB=AA=RC

```

(Continued on top of page 32)

(Continued on next page)

Listing Two (Continued)

```

547 JR IF20
548 BEB! BB,C1
549 IF20 IFNOT! BB,IF21
550 GOSUB! WA
551 JR IF21
552 BEB! AA,NL
553 GOSUB! WA
554 BEB! AA,LF
555 GOSUB! WA
556 IF22
557 IF14
558 -
559 IINC! IMP
560 NDWH3
561 JR
562 BECON! UT,0
563 RET
564 *
565 * Write a byte to output
566 *
567 WRFIL! AA,F2
568 RET
569 *
570 * Get a char from input file
571 *
572 SUBGI
573 RDFIL! CC,F1
574 RET
575 *
576 * Get char & goto LOC88 on end
577 *
578 SUBGC
579 GOSUB! GI
580 BNEB! AA,ER,C0
581 IFNOT! AA,IF22A
582 GOTO! 88
583 BEB! AA,CC,EF
584 IFNOT! AA,IF23
585 GOTO! 88
586 RET
587 *
588 * Converts number to series of ascii digit
589 *
590 SUBSD
591 ISLI! AA,ITU,I00
592 IFNOT! AA,IF24
593 BECON! BB,1
594 JR IF25
595 IEMI! ITU,ITU
596 BECON! BB,0
597 *
598 * Converts number to series of hex digit
599 *
600 SUBSH
601 ISLI! AA,ITU,I00
602 IFNOT! AA,IF26
603 BEB! ND,C1
604 BEB! DS,C0,ZR
605 JR IF27
606 BEB! ND,C0
607 ISLI! AA,I00,ITU
608 IFNOT! AA,NDWH4
609 IDV! IYV,ITU,I10
610 ITMS! IAA,I10,IYV
611 IEMI! IXX,ITU,IAA
612 IEI! ITU,IYV
613 BEI! AA,IXX

```

```

608 BEB! AA,AA,ZR
609 BEB! DS,ND,AA
610 BINC! ND
611 JR WH4
612 NDWH4
613 IF27
614 BEB! DS,ND,OR
615 BEB! ND,ND,BB
616 RET
617 *
618 * Write # into F2
619 *
620 SUBWN
621 WHS
622 GOSUB! SD
623 IEB! IAA,ND
624 ISLI! AA,I00,IAA
625 IFNOT! AA,NDWH5
626 BDEC! ND
627 BEB! AA,DS,ND
628 GOSUB! WA
629 JR WH5
630 NDWH5
631 RET
632 *
633 * Write # to terminal
634 *
635 SUBPN
636 WH6
637 GOSUB! SD
638 IEB! IAA,ND
639 ISLI! AA,I00,IAA
640 IFNOT! AA,NDWH6
641 BDEC! ND
642 BEB! AA,DS,ND
643 WRITE! AA
644 JR WH6
645 NDWH6
646 WRITE! BL
647 RET
648 *
649 * Convert AA as a decimal digit
650 *
651 SUBCD
652 BLEB! BB,ZR,AA
653 BLEB! CC,AA,C9
654 BEB! BB,CC
655 IFNOT! BB,IF28
656 BEB! AA,AA,ZR
657 RET
658 BEB! AA,C0
659 RET
660 *
661 * Convert AA as hex digit
662 *
663 SUBCH
664 BLEB! BB,ZR,AA
665 BLEB! CC,AA,C9
666 BEB! BB,CC
667 IFNOT! BB,IF29
668 BEB! AA,AA,ZR
669 RET
670 BLEB! BB,HA,AA
671 BLEB! CC,AA,HF
672 BEB! BB,BB,CC
673 IFNOT! BB,IF30
674 BEB! AA,AA,HA
675 BEB! AA,AA,CX
676 RET
677 BLEB! AA,C0
678 RET
679 *
680 * (A:TINCMP) ff

```

(Continued on column 2)

**CONSULT YOUR
LOCAL tiny-c DEALER**

Cornerstone Software
P.O. Box 5151
San Jose, CA 95150
800-538-3160 or 408-996-8560

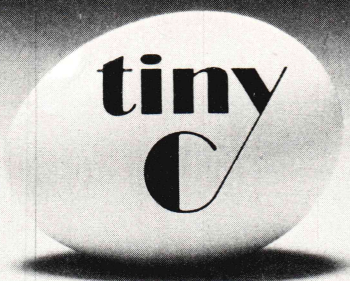
Software Distributors
9929 W. Jefferson Blvd.
Culver City, CA 90026
213-204-6620

SYM - 1 Users' Group
P.O. Box 315
Chico, CA 95927
916-895-8751
(tiny-c ONE Manuals and SYM
cassettes only)

Lifboat Associates
1651 3rd Avenue
New York, NY 10028
212-860-0300

tiny
C

Some things are just
naturally right.



tiny-c is a structured programming language designed to allow you to focus attention on the problems you want to solve — rather than the language you're using to solve it. With **tiny-c** you can expand your horizons far beyond the limits of BASIC. **tiny-c ONE** (interpreter), \$100 - includes Owner's Manual plus wide choice of media, source code. It's still the best structured programming trainer. **Tiny-c TWO** (compiler), \$250 - includes Owner's Manual, CP/M® disk, source code. This version puts UNIX® pleasure into your CP/M.

tiny-c associates

P.O. Box 269, Holmdel, New Jersey 07733 (201) 671-2296

You'll quickly discover **tiny-c** is naturally right for your language needs.

New Jersey residents include 5% sales tax. Visa or Master Charge accepted. Include charge plate number with order.

* CP/M is a trademark of Digital Research, Inc.

* UNIX is a trademark of Bell Labs, Inc.

* **tiny-c** is a trademark of tiny c associates.

**CONSULT YOUR
LOCAL tiny-c DEALER**

Eastern House Software
3239 Linda Drive
Winston-Salem, NC 27106
919-924-2889
(tiny-c ONE Manuals and PET
disks only)

Magnolia Microsystems
2264 15th Avenue W.
Seattle, WA 98119
206-285-7266

VANDATA
17544 Midvale Avenue North
Suite 205
Seattle, WA 98133
206-542-7611

Circle no. 72 on reader service card.

BASIC/Z

the ultimate CP/M* compiler!

- Generates native code (8080/Z-80) for fast execution
- Sort verb is unmatched by stand-alones. 2000 elements in two seconds!
- Alpha-numeric labels, variable and function names of any length
- Chain program segments which share variables declared common
- Five data types - binary/BCD/string
- BCD floating point math - *never* a "round-off" error - precision is program definable from 6-18 digits
- Full function program editor tests syntax as you type
- Recursive, multi-line, multi-argument user defined functions
- Dimension arrays dynamically (to an expression) and selectively erase
- Screen oriented editing of console input at run-time (cursor left/right/start/end, delete left/right/line, insert/change mode, and input masking available)
- Push/pop subroutine stack
- Trace and single-step debugging
- Multi-tiered error trapping even handles BDOS errors
- Cursor addressing, reverse and blinking video, erase and more are supported from source code level, with virtual hardware independence
- An extended library of over 200 "key-word" functions

For free brochure
and mini-manual:

System/z, inc.

P.O. Box 11
Richton Park, IL 60471
(312) 481-8085

a trademark of Digital Research

System/z, inc.

Circle no. 264 on reader service card.

Simplified 68000 Mnemonics

If the 68000 is to become the most important new computer of the next five years, there will necessarily be a tremendous amount of interest in its assembly language. The profusion of features and facilities on the 68000, however, makes it very difficult to get a clear overall view of the instruction set. If one is to program reasonably well in assembly language, one must be able to lay out in one's mind which facilities are there and which are not. With the 68000, it often seems as if one is let loose upon a tropical island, full of flowers of dazzling beauty but so much underbrush that it seems impossible to scale its central mountain peak and survey the totality of the landscape.

The main problem is with the mnemonics. This is not to say that the mnemonics show any glaring examples of bad design. They are serviceable, as far as they go; it is simply that it is possible to do much better. The purpose of this paper is to *propose* a new collection of mnemonics for the 68000. This is not to say that an assembler has been written, using these mnemonics (at the moment the author does not have access to a 68000-based system). Rather, the purpose here is to show the many advantages of this set of mnemonics over Motorola's. There is precedent for such redesigns; for example, the mnemonics of the UNI-VAC 1100 series were redone by CSC in 1962 in connection with CSC's design of the EXEC II operating system for the 1100 series (a project in which this author took part).

The new mnemonics are given in Table I. The notation used in Table I is explained in Table II. It is important to note that the new mnemonics allow the entire instruction repertoire of the 68000 — together with very brief explanations of all instructions — to be presented upon one single-spaced page, with all notational conventions laid out on another single-spaced page.

A program selected at random from a recent book on the 68000¹ is presented in Table III with both the old and new mnemonics. Except for labels, which remain the same in the two cases, the total number of keystrokes of non-blank char-

acters has been reduced from 196 to 98 — a reduction of 50%. The simplicity of understanding for the new mnemonics makes it easy, in fact, to spot a bug in the program of Table III. This will be explained in more detail at the end of the paper.

The basic ideas behind the new mnemonics are as follows:

(1) *The data registers and the address registers all receive new designations.* The data registers are now called the A, B, C, D, E, F, G, and H registers, each of which is 32 bits long. The address registers are

now called the S, T, U, V, W, X, Y, and Z registers, and each of these is 32 bits long. In particular, the S register is the stack pointer, known conventionally as A7. (The original designations for the registers A through H are D0 through D7, and those for S through Z are A7 through A0, in reverse order, with A0 corresponding to Z.)

(2) *There are no suffixes B, W, and L for byte, word, and long word operations.* As we shall see, most instructions do not need such suffixes because they operate on data which has been defined as byte,

G OPERATION	DESCRIPTION	G OPERATION	DESCRIPTION
3 A <u>v</u> , <u>i</u>	Add <u>i</u> to <u>v</u>	3 N <u>v</u> , <u>i</u>	Logical AND <u>i</u> to <u>v</u>
1 Aa <u>v</u>	Add <u>v</u> to <u>a</u> (<u>v</u> ≠ 1t)	6 Na <u>v</u>	Logical AND <u>v</u> to <u>a</u>
ABaB	Add BCD 1b to 1a	NBa	Negate BCD 1a
ABtu	Add BCD 1MDu to 1MDt	4 NBM <u>vl</u>	Negate BCD <u>vl</u> in mem.
4 AMa <u>v</u>	Add <u>a</u> to <u>v</u> (memory)	NCC <u>il</u>	Logical AND <u>il</u> to CC
4 ASd <u>v2</u>	Arith. shift <u>v2</u> by 1	NGma	Negate <u>ma</u> (2's comp.)
ASd <u>ma</u> , <u>sc</u>	Arith. shift <u>ma</u> by <u>sc</u>	4 NGM <u>v</u>	Negate <u>v</u> in memory
1 At <u>vx</u>	Add <u>vx</u> to <u>t</u>	3 NMa <u>v</u>	AND <u>a</u> to <u>v</u> (memory)
AXmaB	Add extended <u>mb</u> to <u>ma</u>	NO	No operation
AXmtu	" " <u>mMDu</u> to <u>mMDt</u>	P NSR <u>i2</u>	Logical AND <u>i2</u> to SR
2 An <u>v</u>	Add <u>n</u> to <u>v</u> (<u>v</u> ≠ 1t)	NTma	NOT <u>ma</u> (ones' compl.)
B <u>p</u>	Branch to <u>p</u>	4 NTM <u>v</u>	NOT <u>v</u> in memory
Baf <u>i2</u>	Test bit <u>i2</u> of <u>a</u> , op <u>f</u>	NXma	Negate extended <u>ma</u>
Bafb	" " <u>y</u> (in <u>b</u>) " " " "	4 NXM <u>v</u>	" " <u>v</u> in memory
BaT <u>i2</u>	Test bit <u>i2</u> of <u>a</u>	3 O <u>v</u> , <u>i</u>	Logical OR <u>i</u> to <u>v</u>
BaTb	Test bit <u>y</u> (in <u>b</u>) of <u>a</u>	6 Oa <u>v</u>	Logical OR <u>v</u> to <u>a</u>
Bcc <u>p</u>	Branch to <u>p</u> on <u>cc</u>	OCC <u>il</u>	Logical OR <u>il</u> to CC
4 Bf <u>vl</u> , <u>i2</u>	Test bit <u>i2</u> of <u>vl</u> , op <u>f</u>	3 OMa <u>v</u>	OR <u>a</u> to <u>v</u> (memory)
4 Bfa <u>vl</u>	" " <u>y</u> (in <u>a</u>) " " " "	P OSR <u>i2</u>	Logical OR <u>i2</u> to SS
BS <u>p</u>	Branch to subroutine <u>p</u>	7 PA <u>v</u>	Push address of <u>v</u>
5 BT <u>vl</u> , <u>i2</u>	Test bit <u>i2</u> of <u>vl</u>	1 PH <u>v</u>	Push <u>v</u> (= M <u>mMDS</u> , <u>v</u>)
BTa <u>vl</u>	Test bit <u>y</u> (in <u>a</u>) of <u>vl</u>	1 PL <u>v</u>	Pull <u>v</u> (= M <u>v</u> , <u>mMSI</u>)
3 C <u>v</u> , <u>i</u>	Compare <u>i</u> with <u>v</u>	PMxa Q(t)	xa+Q(t) alternate bytes
1 Ca <u>v</u>	Compare <u>a</u> with <u>v</u> (<u>v</u> ≠ 1t)	Pxa Q(t)	Q(t)+xa " " (periph.)
6 Cka <u>v2</u>	Check <u>2a</u> bounded by <u>v2</u>	R	Return from subroutine
Cmtu	Compare <u>mMTI</u> with <u>mMuI</u>	RE	" " exceptn. (pull SR)
1 Ct <u>vx</u>	Compare <u>t</u> with <u>vx</u>	8 RM <u>rs</u> , <u>vx</u>	Regs to memory <u>rs</u> → <u>vx</u>
Dcca <u>p</u>	Bcc <u>β</u> ; <u>a</u> -1; <u>a</u> =-1? → <u>p</u> ; <u>β</u> :	RMxt <u>rs</u>	Regs-mem. (=RM <u>rs</u> , <u>xMDt</u>)
6 Dsa <u>v2</u>	Divide, signed <u>4a/v2</u>	ROd <u>v2</u>	Rotate <u>v2</u> by 1
6 DuA <u>v2</u>	Divide, unsigned <u>4a/v2</u>	ROd <u>ma</u> , <u>sc</u>	Rotate <u>ma</u> by <u>sc</u>
Ea	Exchange halves of <u>a</u>	RR	Ret.&restore (pull CC)
Egh	Exchange <u>4g</u> with <u>4h</u>	RS	Reset external devices
FAt <u>-i2</u>	Frame allocation (1Mt)	RXd <u>v2</u>	Rot. extended <u>v2</u> by 1
FDt	Frame deallocation	RXd <u>ma</u> , <u>sc</u>	Rot. extended <u>ma</u> by <u>sc</u>
H <u>i2</u>	Halt (status reg. < <u>i2</u>)	3 S <u>v</u> , <u>i</u>	Subtract <u>i</u> from <u>v</u>
7 J <u>v</u>	Jump to <u>v</u>	1 Sa <u>v</u>	Subtr. <u>v</u> from <u>a</u> (<u>v</u> ≠ 1t)
7 JS <u>v</u>	Jump to subroutine <u>v</u>	SBaB	Subtr. BCD 1b from 1a
7 LA <u>t</u> , <u>v</u>	Load <u>t</u> with addr. of <u>v</u>	SBtu	" " 1MDu from 1MDt
4 Lsd <u>v2</u>	Logical shift <u>v2</u> by 1	3 Scc <u>vl</u>	Set <u>vl</u> = -1 (if <u>cc</u>) or 0
Lsd <u>ma</u> , <u>sc</u>	Logical shift <u>ma</u> by <u>sc</u>	4 SMa <u>v</u>	Subtr. <u>a</u> from <u>v</u> (mem.)
1 M <u>v</u> , <u>w</u>	Move <u>w</u> to <u>v</u> (<u>w</u> ≠ 1t)	1 St <u>vx</u>	Subtract <u>vx</u> from <u>t</u>
1 Ma <u>v</u>	Move <u>v</u> to <u>a</u> (<u>v</u> ≠ 1t)	SXmaB	Subtr. extended <u>ma</u> - <u>mb</u>
6 MCC <u>v2</u>	Move <u>v2</u> (byte 2) to CC	SXmtu	Subtr. ext. <u>mMDt</u> - <u>mMDu</u>
MKa <u>k</u>	Move signed const. <u>k</u> → <u>a</u>	2 Sn <u>v</u>	Subtr. <u>n</u> from <u>v</u> (<u>v</u> ≠ 1t)
1 MMa <u>v</u>	Move <u>a</u> to <u>v</u> (memory)	3 TE <u>v</u>	Test <u>v</u> , set flags N&Z
1 MMt <u>vx</u>	Move <u>t</u> to <u>vx</u> (memory)	TR <u>vn</u>	Trap, using vector <u>vn</u>
9 MR <u>rs</u> , <u>vx</u>	Move <u>vx</u> → registers <u>rs</u>	3 TS <u>vl</u>	TE <u>vl</u> ; <u>vl</u> (hi-order) = 1
MRxt <u>rs</u>	Move regs(=MR <u>rs</u> , <u>xMTI</u>)	TV	BVC <u>β</u> ; TR 7; <u>β</u> :
6 MSA <u>v2</u>	Multiply signed <u>2a*v2</u>	3 X <u>v</u> , <u>i</u>	Exclusive OR <u>i</u> to <u>v</u>
3 MSM <u>v2</u>	Move status reg. to <u>v2</u>	XCC <u>il</u>	Exclusive OR <u>il</u> to CC
P6 MSR <u>v2</u>	Move <u>v2</u> to status reg.	3 XMa <u>v</u>	Excl. OR <u>a</u> to <u>v</u> (mem.)
P MSt	Move <u>t</u> to S	P XSR <u>i2</u>	Exclusive OR <u>i2</u> to SR
1 Mt <u>vx</u>	Move <u>vx</u> to <u>t</u>	Xxa	Extend sign <u>a</u> (to <u>xa</u>)
P MTS	Move <u>S</u> to <u>t</u>	3 Z <u>v</u>	Zero (clear) <u>v</u>
6 MUa <u>v2</u>	Multiply uns. <u>2a*v2</u>		

Table I

by W.D. Maurer

W. D. Maurer, George Washington University, S.E.A.S., Washington, DC 20052.

word, or long word data, which determines the type of the operation automatically. In cases where the length must be given, however, it is given in bytes as 1, 2, or 4. This has the immediate advantage of avoiding the non-standard term "word." (A word has 32 bits, not 16, on the IBM 360 and 370, for example.)

The length designation appears in the following places:

- (a) *In certain of the mnemonics.* The instruction whose Motorola mnemonic is ADDX.L D4,D3 (Add Extended, Long Word, register D4 to register

D3) is simply AX4DE in the new mnemonics. Here D and E denote the D register (formerly D3) and the E register (formerly D4) and the 4 denotes, very simply, "four bytes." Note that the order is "destination first, then source"; see point (3) below to appreciate how much simpler this makes a great number of the mnemonics. The corresponding byte and word comparison instructions ADDX.B D4,D3 and ADDX.W D4,D3 become AX1DE and AX2DE respectively.

- (b) *Preceding register designations.* The instruction MOVE.L A3,A6 (old mnemonics) moves A3 to A6. Renaming these registers W and T respectively, we can write M 4T,4W (reversing the order as before). It is simpler, however, to write MT 4W where MT means "move to the T register" and 4W means "four bytes of the W register." The corresponding byte and word move instructions MOVE.B A3,A6 and MOVE.W A3,A6 become MT 1W and MT 2W respectively.

- (c) *As part of more general addressing modes.* Address register indirect addressing, involving the X register, in a one-byte instruction, would be denoted by 1MX (one byte of memory indexed by X.) The same thing, with auto-increment or auto-decrement, would be 1MXI or 1MDX respectively. (Note that the I, for "increment," follows the register name, whereas D, for "decrement," precedes it; this resembles the placement of + and - in the Motorola mnemonics, and is done for the same reason - because incrementation follows the reference to memory, while decrementation precedes it.) Even the most complex mode, namely based indexed long, is straightforward; thus 2MU4E-9 means "two bytes of memory, indexed by the U register and by all four bytes of the E register, minus the displacement 9." The same thing in Motorola mnemonics would be -9(A5,D4.L) - over 50% more characters (not counting the extra .W on the instruction mnemonic) and lacking the straightforward readability of the new expression.

- (3) *Most operation codes have only one argument.* In fact the only operation codes with two arguments are moves, shifts, and immediate addressing instructions. Thus M KPREV,KCURR sets KPREV (in memory) equal to KCURR (in memory); A FCOUNT,5 adds 5 to FCOUNT in memory; ASL 1H,4 shifts 1H (one byte of the H register, formerly known as D7) left arithmetically by 4. (If the mnemonic variations are counted separately - if MA through MH are counted as eight instructions instead of one, for example - then almost 75% of the mnemonics have no arguments; about 25% have one; and only 2% have two.)

a = A, B, C, D, E, F, G, or H (data registers)
b = A, B, C, D, E, F, G, or H (data registers)
CC = condition code register (rightmost byte of SR)
cc = one of the following forms:

T	true	VC	overflow clear
F	false	VS	overflow set
HI	high	PL	plus
LS	low or same	MI	minus
CC	carry clear	GE	greater or equal
CS	carry set	LT	less than
NE	not equal	GT	greater than
EQ	equal	LE	less or equal

d = L (left shift) or R (right shift)
e = label, label+j, or other address expression (length defined in definition of given label) or mBlabel (length m)
f = N (op=AND with 0), O (op=OR with 1), or X (op=XOR with 1)
g = A, B, C, D, E, F, G, H, S (stack pointer), T, U, V, W, X, Y, or Z (data register or address register); length = 4
h -- same as g (second register in exchange instruction)
i = y (constant) or e (address of e) (immediate data)
i1 -- same as i but length must be 1 (immediate data)
i2 -- same as i but length must be 2 (immediate data)
j = \$hh... or dd... or %bb... where the h's are hexadecimal digits, the d's decimal digits, and the b's binary digits (constant)
k -- +k is +0 through +127; -k is -1 through -128 (constant in move instruction MK; 8-bit displacement in based indexed modes)
m -- 1, 2, or 4 (length, or number of bytes in register or memory)
n -- 1, 2, 3, 4, 5, 6, 7, or 8 (additive or subtractive constant)
p = q3, but *-32766 ≤ p ≤ +32769 (relative address; *this loc.)
q = e (data) or mBj (data, length m, at address j) (memory loc.)
q2 -- same as q but address must have length 2 (memory location)
q3 -- same as q but address must have length 3 (memory location)
r = q3, but *-126 ≤ r ≤ +129 (relative address; *this location)
rs = z1/z2/... where each zk = g or g-h (g, h as above; g-h means registers g, g+1, ..., h) (multiple registers for MR and RM)
sc = 1, 2, 3, 4, 5, 6, 7, 8, A, B, C, D, E, F, G, H (shift count -- in rightmost 6 bits of register A-H if this is specified)
SR = status register (length 2; CC = rightmost byte of SR)
t = S (stack pointer), T, U, V, W, X, Y, or Z (address registers)
u = S (stack pointer), T, U, V, W, X, Y, or Z (address registers)
v = one of the following forms (variable in register or memory):

#i	(immediate)	(Groups 1 and 6 only)
q3	(absolute long)	(All groups)
q2	(absolute short)	(All groups)
ma	(data register direct)	(Groups 1, 2, 3, and 6 only)
mt	(address register direct)	(Groups 1 and 2 only)
mMt	(address register indirect)	(All groups)
mMtI	(post-increment)	(All groups except 7 and 8)
mMDt	(pre-decrement)	(All groups except 7 and 9)
mMtg	or mMtg+k (based indexed short)	(all groups)
mMt4g	or mMt4g+k (based indexed long)	(all groups)
q2(t)	or mMt+q2 (based)	(All groups)
p	(relative)	(Groups 1 and 5 through 9 only)
r(t)	or r(4t) (relative indexed)	(Groups 1 and 5 through 9 only)

vm -- like v but with length m (variable in register or memory)
vn -- 0 through 15 (vector number in trap instruction)
w -- like v in M v,w but group 3 and length must equal length of v
x -- like m, but x ≠ 1 (vx is like v but with length 2 or 4)
y = mBj (length m) or j (length as small as possible) (integer)

Table II

The destination register in a great number of the arguments is incorporated into the mnemonic. For the C register, for example (formerly D2), we have MC (move to C, AC, SC, CC (add, subtract, and compare), MSC, MUC, DSC, DUC (multiply and divide by C, signed and unsigned), NC (AND), OC (OR), and so on. For a few "operate to memory instructions" such as AMC (add to memory), SMC, NMC, and OMC, the C register is the source, rather than the destination. There are also EC (exchange halves of C), ECD (exchange C and D), NG4C (set C to its two's complement), X2C and X4C (extend sign to two bytes or four bytes of C), and other such exceptional cases.

(4) All operations on data have lengths determined by the data unless otherwise directed. If AMOUNT is a 32-bit variable, then MF AMOUNT moves AMOUNT to the F register, and is a long-word (32-bit) operation. To move only one or two bytes of AMOUNT to the F register, one uses MF 1BAMOUNT or MF 2BAMOUNT respectively.

Tables I and II are meant to be used in tandem. As an example, consider the second entry in Table I, namely Aa ν (Add ν to a). By consulting Table II, one learns that a may be A, B, C, D, E, F, G, or H, and that ν may have one of several forms. Let us consider the form mt .

Here m is 1, 2, or 4, but in the description of Aa ν we find the designation " $\nu \neq 1t$ "; that is, ν cannot be of the form $1t$, so that m , here, must be 2 or 4, while t is S, T, U, V, W, X, Y, or Z (again by reference to Table II). Thus there are the following possibilities (for example):

AA 2X
(Add to A from two bytes of X)
AD 4T
(Add to D from four bytes of T)

for a total of 128 ($8 \times 2 \times 8$) possibilities. Another form of ν , in this same instruction, is $q3$, described as "same as q but address must have length 3." So this is a 24-bit (or three-byte) address, and we have the following further possibilities for ν :

$q = e = \text{label}$
(label of location with three-byte address)
 $q = mBj = 4B\$hh...$
(length 4, but address still has length 3)
 $q = e = \text{label} \pm j = \text{label} + dd...$
 $q = e = mB\text{label}$
(data has length m ; address has length 3)

This means that if NST is a two-byte variable, ECOUNT a four-byte variable, and

F7 an array of single-byte variables, we could also write

AF ECOUNT
(Add to four bytes of F from ECOUNT)
AD 4B\$FC
(Add to D from four bytes starting at address \$FC)
AG F7+50
(Add to one byte of G from byte 50 of F7)
AA 1BNST
(Add to one byte of A from one byte of NST)

Still another form of ν , in this same instruction, is $\#i$, for immediate data. By Table II, we can have $i = y = mBj = 4B\$hh...$ which gives us the possible instruction

AD #4B\$FC
(Add \$FC to four bytes of D)

Note the two meanings of the 4 in 4B\$FC — in this instruction it means "the constant \$FC as a four-byte constant," whereas in AD 4B\$FC it means "four bytes of memory."


The column headed G in Table I is the group. Instructions belong to groups, and each group can use only certain of the ν options as specified in Table II. For example, the instruction LA $t \nu$ belongs to group 7. This means that ν can be of

Elegance

Power

Speed



BDS  Users' Group
Supporting All C Users
Box 287
Yates Center, KS 66783

<u>OLD</u> <u>MNEMONICS</u>										<u>NEW</u> <u>MNEMONICS</u>															
PGM_9_4A	MOVEA.L	LIST,A0								PGM_9_4A	MZ	LIST													
	CLR.W	D0									Z	2A													
	MOVE.B	(A0)+,D0									MA	1MZI													
	LEA	-1(A0,D0.W),A1									LAY	1MZA-1													
SORT	CLR.W	D1								SORT	Z	2B													
	MOVEA.L	A0,A2									MX	4Z													
NEXT	MOVE.B	(A2)+,D0								NEXT	MA	1MXI													
	CMP.B	(A2),D0									CA	1MX													
	BCC.S	NSWITCH									BCC	NSWITCH													
	MOVE.B	(A2),D1									MB	1MX													
NSWITCH	MOVE.B	D1,-1(A2)								NSWITCH	MMB	1MX-1													
	MOVE.B	D0,(A2)									MMA	1MX													
	ADDQ.W	#1,D1									A1	2B													
	CMPLA.L	A2,A1									CY	4X													
	BHI	NEXT									BHI	NEXT													
	TST.W	D1									TE	2B													
	BNE	SORT									BNE	SORT													
	RTS										R														
<u>REGISTERS</u>																									
<u>OLD</u>	D0	D1	D2	D3	D4	D5	D6	D7	A0	A1	A2	A3	A4	A5	A6	A7									
<u>NEW</u>	A	B	C	D	E	F	G	H	Z	Y	X	W	V	U	T	S									

Table III

the form $r(t)$ or $r(4t)$, since these are specified in Table II as "groups 1 and 5 through 9 only." Thus if WTABLE is a suitable relative address, then each of the following is *legal*:

LAT WTABLE(Z)

(Load address of WTABLE plus Z into T)

LAV WTABLE(4W)

(Load address of WTABLE plus W into V)

where two bytes of Z, and four bytes of W, are used. On the other hand, v cannot be of the form $mMtI$ or $mMDt$, since these are specified in Table II as "all groups except 7 and 8" or "all groups except 7 and 9." Thus each of the following is *illegal*:

LAT 2MZI

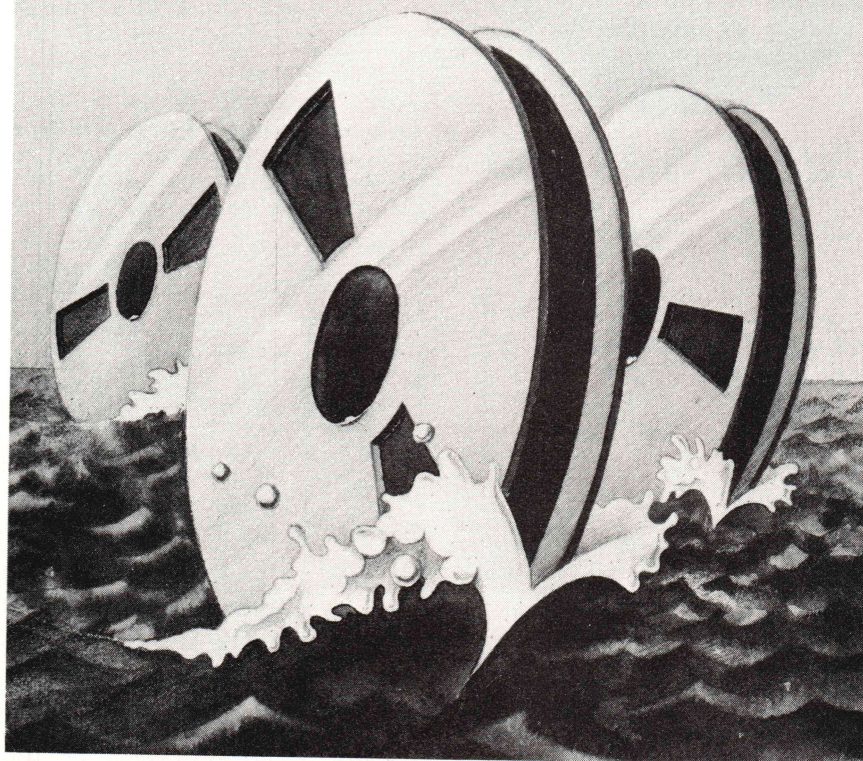
LAV 4MDW

It should be noted that the length (1, 2, or 4) is meaningless in the case of the specific instructions LA_t , PA_t , J, and JS. Thus in the program of Table III, LAY 1MZA-1 means "Load Address, into Y, of the memory word indexed by Z and by A, minus 1." The 1 before the M means nothing, but it must be present (although 2 or 4 would do just as well), since LAY MZA-1 would load the address of a variable called MZA, minus 1. (Privileged instructions are group P.)

Examples of the v options which start with a length (1, 2, or 4) are as follows:

- (1) 1C (one byte of the C register; the other three bytes are ignored)
- (2) 2V (two bytes of the V register; extended through the other two bytes)
- (3) 1MU (one byte in memory, indexed by U — that is, the address of this byte is contained in U)
- (4) 2MWI (two bytes in memory, indexed by W, and then increment W by 2 after making the reference)
- (5) 4MDT (four bytes in memory, indexed by T, and decrement T by 4 before making the reference)
- (6) 2MUE (two bytes in memory, indexed by U and by E — that is, the address of this byte is the sum of U and two bytes of E)
- (7) 1MWX (one byte in memory, indexed by W and by X)
- (8) 4MZ4G (four bytes in memory, indexed by Z and all four bytes of G)
- (9) 4MXY+8 (four bytes in memory,

VICTORY AT C.



The Z-80 C Cross-Compiler Fleet has arrived.

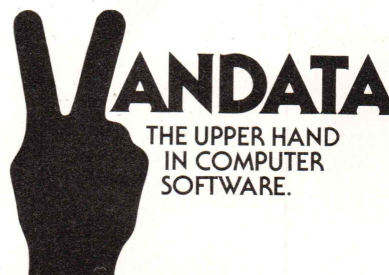
Is your Z-80 C object code larger than it needs to be? Retreating to assembler because C isn't fulfilling its promise?

Your ship has come in. Vandata introduces the Z-80 C Cross-Compiler. Starting with the finest "pure" C-to-8080 cross-compiler available (Whitesmiths), we've added 8-bit expressions, a tight calling convention, a peephole optimizer and full use of the Z-80 instruction set to reduce your code size by up to 36%.

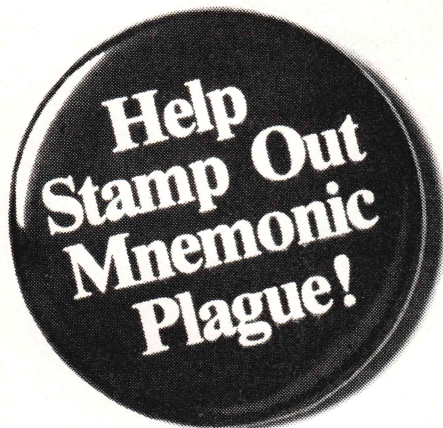
And if you're in a ROM-based environment, you'll appreciate our in-line machine code, listings, ROM-able tables and optional royalty-free runtime library!

The Vandata Z-80 C Cross-Compiler is currently available for VAX, PDP-11 and ONYX hosts.

For more information call
(206) 542-7611 or write VANDATA,
17544 Midvale Ave. N., Suite 205,
Seattle, WA 98133.



End the Dark Ages of
Assembly Language....



with **SMAL/80**

SMAL/80	Assembler
B=C	MOV B,C
D=3	MVI D,3
A=M(HL)	MOV A,M
A=M(DE)	LDAX D
M(DE)=A	STAX D
HL=M(L6)	LHLD L6
HL=5	LXI H,5

SMAL/80 gives you the logical power, versatility and convenience of a compiled, structured high level language like Pascal, Ada or C, plus the efficiency of assembly language.

☐ intuitive, processor-independent symbolic notation system to make your programs easy to read, debug and maintain;

☐ programming constructs BEGIN...END, IF...THEN...ELSE, and LOOP...REPEAT, plus indentation, to graphically display the structure of your algorithms;

☐ extremely flexible macro and text pre-processor to create your own programming environment;

☐ compiler/linker to mix your input source code and relocatable object code, creating modular programs;

☐ translator program to automatically upgrade your assembly code to SMAL/80;

☐ available on CP/M disks with manual for \$150 plus \$4 shipping.

☐ Macro processor available separately for \$75. Complete tutorial text: "Structured Microprocessor Programming" (Publ. Yourdon Press) \$20.

Send for your free button and literature or try the Ultimate Demo: SMAL/80 is 100% Guaranteed! Return it for a full refund within 45 days of purchase if not satisfied.

Chromod Associates,
1030 Park Ave., Hoboken, N. J. 07030
Telephone: (201) 653-7615

Also available from
WESTICO (203) 853-6880

indexed by X and by Y, plus 8 — that is, the address of this byte is the sum of X and two bytes of Y, plus 8)

- (10) 1MZ4S-16 (one byte in memory, indexed by Z and by all four bytes of the stack pointer S, minus 16)

Table IV gives examples of all the instructions in both the old and the new mnemonics; Table V gives a more expanded explanation of the program of Table III. We may note from Table V that the use of the new mnemonics makes it much

easier to spot a bug in this sorting program, intended to sort a table of unsigned, single-byte quantities into descending order. The B register (formerly D1) is being used as the exchange flag: set to zero (Z 2B), incremented to show the exchange (A1 2B), and tested at the end TE 2B). But it is also used as a temporary register in moving 1MX to 1MX-1 (in the instructions MB 1MX and MMB 1MX-1). If the bug is not fixed, the instruction A1 2B might increment -1 to zero, making it look like the exchange flag is still clear.

MNEMONIC	EXAMPLE	MOTOROLA MNEMONIC	MNEMONIC	EXAMPLE	MOTOROLA MNEMONIC
A v, i	A 2MZ1,1	ADD.W #1,(A0)+	N v, i	N 4MDY,8	AND.L #8,-(A1)
Aa v	AA 4MDY	ADD.L -(A1),D0	Na v	NB 1MXI	AND.B (A2)+,D1
ABab	ABGH	ABCD D7,D6	NBa	NBC	NBCD D2
ABtu	ABWX	ABCD -(A2),-(A3)	NBM v1	NBM 1MZT	NBCD 0(A0,A6)
AMa v	AMB BYT1	ADD.B D1,BYT1	NCC i1	NCC \$1A	AND.B #\$1A,SR
ASd v2	ASR WD1	ASR WD1	NGma	NG1A	NEG.B D0
ASd ma, sc	ASL 2C,4	ASL.W 4,D2	NGM v	NGM 2MUD	NEG.W 0(A5,D3)
At vx	AY 4MV	ADD.L (A4),A1	NMa v	NMD 4MDS	AND.L D3,-(A7)
AXmb	AX1DE	ADDX.B D4,D3	NO	NO	NOP
AXmtu	AX2UZ ADDX.W -(A0),-(A5)		NSR i2	NSR \$FOF	AND.W #\$FOF,SR
An v	A4 4F	ADDQ.L #4,D5	NTma	NT2E	NOT.W D4
B p	B LABEL1	BRA LABEL1	NTM v	NTM BYT1	NOT.B BYT1
Baf i2	BGN 8	BCLR #8,D6	NXma	NX4F	NEGX.L D5
Bafb	BHOF	BSET D5,D7	NXM v	NXM 1MS	NEGX.B (A7)
BaT i2	BET \$10	BTST \$10,D4	O v, i	O 2G,\$AA	OR.W #\$AA,D6
BaTb	BDTC	BTST D2,D3	Oa v	OH BYT1	OR.B BYT1,D7
Bcc p	BEQ BETA	BEQ BETA	OCC i1	OCC 8	OR.B #8,SR
Bf v1, i2	BX M9,6	BCHG 6,M9	OMA v	OMB 4MUI	OR.L D1,(A5)+
Bfa v1	BNB 1MTI	BCLR D1,(A6)+	OSR i2	OSR \$200	OR.W #\$200,SR
BS p	BS PROG3	BSR PROG3	PA v	PA 1MV	PEA (A4)
BT v1, i2	BT M8,3	BTST 3,M8	PH v	PH WD1 MOVE.W WD1,-(A7)	
BTA v1	BTA 1MDZ	BTST D0,-(A0)	PL v	PL 4A MOVE.L (A7)+,D0	
C v, i	C 4MSI,5	CMP.L #5,(A7)+	PMxa Q(t)	PM2B J(Z)	MOVEP.W D1,J(A0)
Ca v	CD 1A	CMP.B D0,D3	Pxa Q(t)	P4C J(Y)	MOVEP.L J(A1),D2
CKa v2	CKB WD1	CHK WD1,D1	R	RTS	
Cmtu	C4YZ CMPM.L (A0)+,(A1)+		RE	RTE	
Ct vx	CX 2W	CMP.W A3,A2	RM rs, vx	RM T,LW1	MOVEM.L A6,LW1
Dcca p	DGCS L19	DBCS D2,L19	RMxt rs	RM4Z T-Y	" A1-A6,-(A0)
Dsa v2	DSD WD1	DIVS WD1,D3	ROd v2	ROR WD1	ROR WD1
DUA v2	DUE 2MU	DIVU (A5),D4	ROd ma, sc	ROL 1D,C	ROL.B D2,D3
Ea	EF	SWAP D5	RR	RTR	
Egh	ESA	EXG A7,D0	RS	RESET	
FAt -i2	FAZ -23	LINK A0,-23	RXd v2	RXL WD1	ROXL WD1
FDT	FDT	UNLK A6	RXd ma, sc	RXR 4E,6	ROXR.L 6,D4
H i2	H \$FOFO	STOP \$FOFO	S v, i	S 1MW,32	SUB.B #32,(A3)
J v	J 1MYD-8	JMP -8(A1,D3)	Sa v SA	2MX4G-8 SUB.W -8(A2,D6.L),D0	
JS v	JS 1MX4G	JSR (A2,D6.W)	SBab	SBFE	SBCD D4,D5
Lat v	LAW 1MVI	LEA (A4)+,A3	Sbtu	SBZY	SBCD -(A1),-(A0)
Lsd v2	LSL WD1	LSL WD1	Scc v1	SGT BYT1	SGT BYT1
Lsd ma, sc	LSR 1F,H	LSR.B D7,D5	SMA v	SME 4MWH	SUB.L D4,0(A3,D7)
M v, w	M LW1,J3	MOVE.L J3,LW1	St vx	SU 2F	SUB.W D5,A5
Ma v	MC 2,\$#C	MOVE.B #5C,D2	SXmb	SX4HB	SUBX.L D1,D7
MCC v2	MCC \$1A	MOVE \$1A,CCR	SXmtu	SX1US	SUBX.B -(A7),-(A5)
MKa k	MKH -100	MOVEQ -100,D7	Sn v	S7 2MTI	SUBQ.W #7,(A6)+
MMa v	MMA WD2	MOVE.W D0,WD2	TE v	TE 1MV4X	TST.B 0(A4,A2.W)
MMt vx	MMV 4MU	MOVE.L A4,(A5)	TR vn	TR 4	TRAP 4
MR rs, vx	MR T,LW1	MOVEM.L LW1,A6	TS v1	TS 1G	TAS D6
MRxt rs	MR4Z A/H	" (A0)+,D0/D7	TV	TV	TRAPV
MSa v2	MSB 2MDW	MULS -(A3),D1	X v, i	X 1MW,\$FF	EOR.B #5FF,(A3)
MSM v2	MSM 2MVI	MOVE SR,(A4)+	XCC i1	XCC \$2	EOR.B #2,SR
MSR v2	MSR 2MU	MOVE (A5),SR	XMa v	XMD LW1	EOR.L D3,LW1
MSt	MSX	MOVE A2,USP	XSR i2	XSR \$400	EOR.W #\$400,SR
Me vx	MY 2MZ	MOVE.W 4,(A1)	Xxa	X4C	EXT.L D2
MeS	MUS	MOVE USP,A5	Z v	Z 1MZ4Y+2	CLR.B 2(A0,A1.W)
MUa v2	MUC WD3	MULU WD3,D2			

Table IV

An example which shows the bug is that of a table of size greater than \$82 (that is, hexadecimal 82), which is already sorted in descending order except that its last two bytes are \$FF (and no other byte is \$FF). In the given program, these final two bytes move up by one position in each pass through the table; and only two exchanges (the ones involving these two bytes) are made in each pass, since the table is otherwise already sorted in the proper order. Therefore, MB and A1 are executed only twice in each pass through the table. In the first pass, \$FF is moved into B and this is then incremented by 1 as a two-byte quantity, producing \$100. Subsequently, \$FF is moved into the *rightmost byte* of B, leaving the rest of B undisturbed; this produces \$1FF, which is then incremented to \$200. Thus the final value of B is \$200 at the end of the first pass, and, in general, it is 2 x \$k00 at the end of the kth

pass. At the end of the 128th pass, therefore, B will be zero; the TE then tests for zero; the BNE does *not* branch back; and the sort terminates with the two \$FF bytes in positions 129 and 130 from the end, instead of in the first two positions, where they belong.

References

- ¹ Kane, G., D. Hawkins, and L. Leventhal, *68000 Assembly Language Programming*, Osborne/McGraw-Hill, Berkeley, 1981.

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 524

PGM_9_4A	MZ	LIST	(Move LIST, which is a 4-byte quantity, to the Z-register)
	Z	2A	(Zero, or clear, two bytes of the A-register)
	MA	1MZI	(Move, to the A-register, one byte of memory, indexed by the Z-register, and then increment the Z-register by 1)
	LAY	1MZA-1	(Load into the Y-register the address in memory indexed by the Z register and by the A register, minus 1)
SORT	Z	2B	(Zero, or clear, two bytes of the B-register)
	MX	4Z	(Move, to the X-register, four bytes of the Z-register)
NEXT	MA	1MXI	(Move, to the A-register, one byte in memory, indexed by the X-register, and then increment the X-register by 1)
	CA	1MX	(Compare the A-register with one byte in memory, indexed by the X-register)
	BCC	NSWITCH	(Branch on carry clear to NSWITCH)
	MB	1MX	(Move, to the B-register, one byte in memory indexed by the X-register)
	MMB	1MX-1	(Move to memory, from the B-register, to one byte in memory, indexed by the X-register minus 1)
	MMA	1MX	(Move to memory, from the A-register, to one byte in memory, indexed by the X-register)
	A1	2B	(Add 1 to two bytes of the B-register)
NSWITCH	CY	4X	(Compare the Y-register with four bytes of the X-register)
	BHI	NEXT	(Branch on high, i. e., Y > X, to NEXT)
	TE	2B	(Test two bytes of the B register)
	BNE	SORT	(Branch to SORT on unequal to zero)
	R		(Return from subroutine)

REGISTERS

OLD	D0	D1	D2	D3	D4	D5	D6	D7	A0	A1	A2	A3	A4	A5	A6	A7
NEW	A	B	C	D	E	F	G	H	Z	Y	X	W	V	U	T	S

Table V

Now! A periodical dedicated exclusively to the needs of the CP/M user

CP/M Review.

Each of the six bimonthly issues brings you comprehensive news of the public domain and commercially available software. Features, articles and CP/M news.

Special Offer

CP/M Review's first installment of the on-going in-depth Buyers Guide for \$5.00. This includes a handy reference card and shipping costs.

The Buyers Guide covers over fifty CP/M systems, including configuration, cost, peripherals, software and service.

The above offer is included free when you subscribe to CP/M Review

ENTER YOUR ORDER TODAY

☐ Yes. Enter my subscription for 1 year (6 issues) which includes the above offer. Enclosed is \$18.00*

☐ Enclosed is \$5.00* for my Buyers Guide and CP/M reference card only.

Name _____

Address _____

City _____

State _____ Zip _____

Mail to

CP/M Review
2711 76th Avenue S.E.
Mercer Is, WA 98040
(206) 232-6719

CP/M is the trademark of Digital Research Inc.
*Washington residents add 6.3% Sales Tax.

CP/M REVIEW

THE PUBLICATION FOR THE CP/M COMMUNITY

CP/M EXCHANGE

by Gene Head

Mail received after the first CP/M Exchange was encouraging. There is heavy interest in modem communication and program exchange. The most often asked question was, "How do I get started using my modem?" Well, it may be as simple as one, two three. I say *may* because each modem installation is different. If you have a basic understanding of the following three steps, you should be able to get signed-on to a Remote CP/M (RCP/M) System using the MBOOT3 program (listing begins on page 47).

1. Modem Status
2. Modem Input
3. Modem Output

If you understand how the CP/M functions CONSOLE STATUS, CONSOLE INPUT and CONSOLE OUTPUT operate, you also understand how the modem should operate. (Note: CONSOLE routines use only seven bits of data and the MODEM routines *must* use all eight bits.) Don't be put off if you fail

to understand how to get the CP/M CONSOLE functions to operate. You'll need special assistance to get your modem functional, but I have a plan for you too!

MBOOT3 is a simple, stripped-down, receive-only version of Ward Christensen's MODEM program. The idea is to get MBOOT3 operating and use it to download (receive) a full MODEM program from an RCP/M system. If you have a modem, the software to simulate a terminal *and* the ability to "capture" received data into a file, just log on to an RCP/M and command the remote computer to list the MBOOT3 file using the CP/M command TYPE. Otherwise, spend an hour or so and type in MBOOT3.

Customize the hardware-dependent equates, assemble, and load MBOOT3. Follow the instructions on the facing page, remembering that each RCP/M system may be slightly different. Use your head. Don't be discouraged. Try to get the SYS-

tem Operator's help by using the CHAT command, or try ringing his bell by typing control-G. You can't hurt the remote system, and every RCP/M I've ever used has been very forgiving of errors and even untimely disconnections.

Modem Hint # 1

You will probably need cooperation from someone with a modem to test your system out before making long-distance calls to RCP/M's. If you can't get that special help, consider this: colleges with computer departments usually have a modem telephone for remote users. Without an account number and/or password you can't do much but you can at least verify that the basic communication, modem-to-modem, is operating. Execute the MBOOT3 program as directed in the source listing and then access any modem line by telephone. Try a few carriage returns to get the remote system "talking" to your system. Words like FULL and

PROBLEM:

How to mate your CP/M® or ISIS® system to the
Z-8000

(without losing all your present capabilities)

SOLUTION:

ZEX is the software connection which permits Z-8000 applications to run under your current operating system, resulting in portable 16-bit software which runs under CP/M or ISIS! And ZEX is user-configured for any Z-8000 alternate bus master, so that it can support your prototype hardware, as well as commercially available Z-8000 boards.

ZAS is a complete Z-8000 Development Package, which includes a powerful relocatable cross assembler (ZAS), a flexible object task builder (ZLK), and an absolute object loader (ZLD), as well as the ZEX run time module. The package was developed specifically for the Z-8000, and supports the complete Zilog instruction syntax for both the Z8001 and Z8002, including support for mixed segmented and non-segmented code.



**WESTERN
WARES**

Box C Norwood, CO 81423 (303)327-4898

ZAS Package CP/M .. \$395
ISIS-II .. \$495

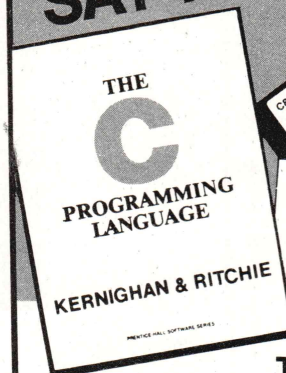
Manual Only..... \$25

Supplied on Single Density 8" Disk

CP/M: Digital Research, Inc. ISIS-II: Intel Corp.

**THEY
SAY IT ALL...**

**WE
DO IT
ALL!**



ANNOUNCING THE C86™ C COMPILER —THE COMPILER THAT SPEAKS THE LANGUAGE OF THE FUTURE!

Kernighan and Ritchie's book, *The C Programming Language*, is the key source for C. Just as fundamental is the C86™ C Compiler.

The C86™ C Compiler is especially designed for the IBM® Personal, IBM® Display Writer, CP/M-86® and MS-DOS®

For further information on the C programming language and the C86™ C Compiler, please contact:



Computer Innovations, Inc.
75 Pine Street
Lincroft, New Jersey 07738
Telephone: (201) 530-0995

C86 is a trademark of Computer Innovations, Inc.; CP/M-86 is a trademark of Digital Research; IBM and MS-DOS are registered trademarks of International Business Machines, Inc.

Circle no. 18 on reader service card.

HALF and HELP might get results. Check it out. . .

Special Assistance

Here is where the novice can be helped by the experienced modem user. If you have the MODEM program working on your system, send me a card with all the details about your specific hardware, the source listing of *just the hardware-dependent equates* and any unique hardware-dependent code, and how they are configured for your set-up.

Likewise, if you have the hardware but don't know exactly how to customize the software, send me a card with your specific hardware specifications. (Keep it simple!) I'll match your request with a known working system and send you the customization details. If I can't get an exact match, I'll send along the best data I have available. *Be sure to enclose an SASE with your request.* I'll do free photocopying (but no free postage) as long as this doesn't get too involved. . .

If you can offer help or need help, let the "CP/M Exchange" know! Be sure to include your disk format and complete modem configuration. Send your complete name, address, city, state, zip, and phone with area code.

Exchange Mail

The first response to the CP/M Exchange came from DDJ contributor Bob Blum. Bob suggested electronic mail for communication. If the interest is there, we might be able to compile a directory of those using this form of communication. I will make my RCP/M system available if there is serious interest. Anyone else have ideas along these lines? Bob is also preparing a contribution on keyboard buffering in a modified CCP for CP/M that will be included in a future "CP/M Exchange."

The most specific letter seeking help came from Charles Henderson of Midland, Texas. Charles is interested in why SPEED and FAST won't work with CP/M 2.x. He suspects the track buffering is involved but doesn't understand all the implications. A definitive article on track and sector skewing, interleaving, sector tables, etc. would be most appropriate. If you have a solid understanding of these principles, write a paper and send

it in! Share with others what you have learned.

Dana Trout of Goleta, California, notes that CP/M Function #37 does not work as described in the manual. This function to reset the disk drives will reset all drives *except* the current default drive. If the current default drive is set to R/O and function thirty-seven is executed, the default drive is still in R/O! Anyone have a fix and/or explanation for this?

Roland Lupient, KB9RR, has ham radio gear, an Altair 8800 and CP/M User's Group volume 41. He would like to hear from anyone who can help interface the ham gear and computer (address: 1953 Graham Lane, Mosinee, WI 54455).

Finally, William Burnett of Sinton, Texas, wonders if there are any Super-

brain users out there who have access to the CP/M User's Library? The same question can be asked about North Star, Osborne, and all the other five-inch formats. If you can copy the CP/M User's Library to any of the many five-inch disk formats, let me know. I suspect you will be in great demand, and while you may not make a lot of money at this, a couple of bucks copying fee seems reasonable.

That covers it for now. Every letter to the "CP/M Exchange" has received either a personal response or was noted in this column. If you have something to contribute, a question or comment, let's hear from you!

DDJ

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 528

Downloading MODEM Using MBOOT3

First, execute MBOOT3 as instructed in the source listing (begins on page 47). It will sign on, but there will be no screen activity after the sign-on messages. In the TERMINAL mode, all displayed console characters come from the remote system. Even keys from your keyboard are "echoed" back, so if there is no remote system to echo back the keystrokes, you won't see them on your CRT.

Now, call a remote CP/M System (see list on page 44), when you hear the remote modem carrier tone, connect your modem to the phone line. The two modems are now connected and your computer is "talking" to the remote computer. Press the Return key once a second until you see the remote computer's sign-on message, usually asking how many nulls you need. (Answer 0 to the nulls question if you're using a video terminal.)

Follow the host computer's instructions until you finally reach the CP/M operating level with the expected A>. You are now in control of the remote machine.

To find the MODEM program type:

DIR MODEM *.*

This will show you the exact name of the MODEM files available. You should see in the listing the MODEM.ASM file or some similar source code file.

To receive the file type:

XMODEM filename.type

The remote system should tell you that XMODEM is ready to send the requested file, then it will wait. There will be no further screen activity from the host computer. It will look as though you have lost control, but don't fret. The host is waiting for your MBOOT3 program to send special protocol characters to begin the transfer and won't respond to your keyboard characters until the transfer is completed or aborted.

Now that you know the host is ready to send and is waiting for you to be ready to receive, press your ESCape key. This key tells MBOOT3 to receive the file the host is sending.

Most CP/M commands are available to remote users, so use the remote system just as you would your local computer. Be sure to sign off by typing BYE and disconnect your modem from your phone line, if it isn't automatically disconnected by your software.

As with any new operation, practice and experience are the best teachers.

Remote CP/M Software Exchange Systems List # 24

Last Revised March 21, 1982

List #24 revised and updated courtesy of Hyde Park RCPM, CP/M-Net, and Mississauga RCPM.

This is part of a CP/M file called RCPMLIST.xx. It is found on most RCP/M systems. The systems presented here are representative of what you can expect to find. Each was "up" as of February, 1982, unless otherwise noted. Log on to any system and then get a copy of this complete file for an accurate directory.

This is a partial list.

California

(Best)

Bakersfield, CA

CP/M-Nettm, (805) 527-9321, Kelly Smith
No A.L.D.S.; 1900-2300 (PST) Monday-Friday,
1900 Friday to 0700 Monday
110-600+ baud
20 Mb of files on 2 hard disks (=8 logical disks)
System now includes SIG/M Vol. 1-10 = E.; SIG/M
Vol. 11-20 = F.; SIG/M Vol 21-25 = G.; XMODEM
'DISKMENUE.DOC' for entire system directory (over
2100 files now available!).

Garden Grove, CA

G.F.R.N. Data Exchange RBBS, (714) 534-1547,
Doug Laing
24 hour operation
300 and 1200 baud
5 Mb of files on 4 drives
Special interest in amateur radio and Apple/CPM. This is
the second G.F.R.N. system.

Larkspur, CA

Larkspur RBBS/RCPM (415) 461-7726, Jim C.
24 hour operation
Up as of 3/20/81
110, 300, 450, 710 baud; SPRINT, ITT, MCI
2+ Mb on 2 drives
The system carries general and new CP/M software. System
now running MP/M and plans are afoot to install a
second telephone line, making it the first multi-user
remote system. SYSOP will assist others in bringing up
MP/M remotely.

Palos Verdes, CA

G.F.R.N. Data Exchange RBBS, (213) 541-2503,
Skip Hansen
24 hour operation
300 and 1200 baud; SPRINT, MCI, ITT
2.4 Mb of files on 2 drives
Standard CP/M software with special interest in ham radio-
related programs. Soon (with MP/M) will also be reach-
able through 450 MHz radio.

San Diego, CA

San Diego RCPM, (714) 271-5615, Brian Kantor
No call back; 24 hour operation
300 and 1200 baud; ITT, SPRINT, MCI
2.4 Mb of files on 2 drives
Now with 212-compatible modem; yet another 1200-baud.

East Central

Allentown, PA

Allentown RBBS/RCPM System, (215) 398-3937,
Bill Ernest
No call back; 24 hour operation
110, 300, 450, 600, 710 baud; SPRINT and ITT
4.25 Mb of files on hard disk (=4 logical disks)
General CP/M software. Bulletin board of the Lehigh
Valley Computer Club.

Grafton, VA

Grafton, VA RBBS, (804) 898-7493, Dave Holmes
No call back, no A.L.D.S.; 24 hour operation
300 baud
200 Kb of files on 2 drives
Carries CP/M, TRS-80 and Apple software; plans for
setting up a dual system (on one line) with an LNW-80
as well as the CP/M computer. Active as bulletin board.

McClean, VA

RLP RCP/M, McClean, VA, (703) 524-2549, Bob Plouffe
No call back; 24 hour operation
SPRINT and MCI
4 (N*) drives with 640 Kb of files
Running CBBS for messages. New system.

Midwest

Chicago (area), IL

Calamity Cliffs Computer Center, (312) 234-9257
No call back; 1400-0200 daily
300, 450, 600 baud; ITT, SPRINT, MCI
11 Mb of files on a hard disk and 2 floppies
Many of the CPMUG and SIG/M programs available by
request.

Chicago, IL

HUG-CBBS, (312) 671-4992, Paul Mayer, Dave Leonard
No call back; 2300 to 1900, 7 days/week
300 baud; SPRINT, ITT, MCI
2 Mb of files on 2 drives
H89-based, operated for the Heath-Zenith Users' Group
and with a special interest in H19- and H89-adapted
(as well as general CP/M) software.

Hyde Park, IL

Hyde Park RCPM/RBBS, (312) 955-4493, Ben Bronson
No call back; 0100-1700 daily
110, 300, 450, 600, 710 baud; SPRINT, ITT, MCI
2 Mb of files on 2 drives
Special interest in hardware and software reviews, C pro-
grams, and very recent releases of standard programs.
SYSOP now testing substantial upgrade of RBBS
programs.

Royal Oak, MI

Royal Oak CP/M, (313) 759-6569, Keith Petersen

Call back; 24 hour operation

110, 300, 450, 600 baud — 1200 baud modem now available on request; use CHAT or leave a message if you want the 212A switched in — ITT, SPRINT, MCI
600 Kb on two floppy drives and 10 Mb on hard disk (=2 logical drives)

Emphasis on new programs and recent updates of standard programs.

Westland, MI

Westland, MI RBBS/RCPM, (313) 729-1905, Ron Fowler
Call back; 24 hour operation
110, 300, 450 and 600 baud; SPRINT, MCI, ITT
1.4 Mb of files on 2 drives
Emphasis on very recent releases.

Northeast

Lexington, MA

Superbrain RCPM, (617) 862-0781, Paul Kelly
1900-0700 Weekdays, 24 hours weekends
110, 300, 1200 baud; SPRINT, ITT, MCI
300 Kb files on-line
Special interest in Superbrain-adapted CP/M programs.

Bearsville, NY

Bearsville Town SJBBS, (914) 679-6559, Hank Szyszka
No call back, no A.L.D.S.
110, 300, 450, 600, 710 baud
2 Mb of files on 4 drives
Installing MP/M. All CP/MUG programs available by request. General CP/M software.

Rochester, NY

Rochester RBBS, (716) 223-1100, Arnie McGall
No call back; 24 hour operation
110 and 300 baud; SPRINT, MCI, ITT
1.8 Mb of files on 3 drives
S-100 based. General CP/M software. The standard RBBS/RCPM system co-exists with a separate passworded message system called DataStar, which can be entered from CP/M but runs on a separate computer. 600 baud capability expected soon.

Hamilton, ON

Hamilton Area Packet Radio Network (HAPN), (416) 335-6620, Stu Beal
No A.L.D.S.; 24 hour operation
110, 300, 450, 600, 710 baud
@? Kb files on-line
New system. System is also linked to radio network and may be accessed via ham radio. Special interest in radio software.

Toronto, ON

Mississauga, Ontario RCPM, (416) 826-5394, Jud Newell
No A.L.D.S.; 24 hour operation
110, 300, 450, 600, 710, 1200 baud
20 Mb hard disk now on-line 24 hours a day
1200 baud Vadic/Bell 212A standard both supported.
300/1200 baud modem available Monday-Friday, PMMI weekends. XMODEM, DISKMENU.DOC and MAST.CAT for details of over 3000 available files.

(Continued on page 46)

Q/C Leads the Pack.



Q/C leads the pack of C compilers for CP/M. For only \$95 you get an excellent compiler that is fully supported. And Q/C includes the *full source code* to the compiler! The 88-page manual sets standards for readability and clarity. (There is even a chapter on compiler internals.)

Get in front of the pack: write for details of the new Version 1.1 of Q/C.

**THE CODE
WORKS**

5266 Hollister
Suite 224
Santa Barbara, CA 93111
(805) 683-1585

CP/M is a trademark of Digital Research.

Circle no. 102 on reader service card.

AT LAST!
A PROFESSIONAL JOURNAL FOR ENGINEERS
SCIENTISTS MATHEMATICIANS & STATISTICIANS USING
MICROCOMPUTERS.

PLUG INTO...

ACCESS!

The Journal of Microcomputer Applications
for

- * numerical analysis
- * math modeling
- * statistical analysis
- * computerized design
- * process simulation
- * report generation

The articles in ACCESS are written by working engineers and scientists who share their knowledge of how to make productive use of microcomputers with you. Your subscription to ACCESS will make your microcomputer more useful in all areas where engineers and scientists use microcomputers. And you'll even find ways to use your computer you hadn't thought of. The articles in ACCESS are written with you in mind and are aimed at helping you turn your microcomputer into the most productive tool possible. Sign up NOW be a charter subscriber. Join the other engineers and scientists who make ACCESS their source of information on microcomputer applications. Charter rates are 6 issues for \$16. (Canada & Mexico \$20. Other \$32). Fill out the coupon below TODAY. Send check, money order, purchase order, or use your VISA or MASTER CARD.

(Sign me up. \$16 ☐ enclosed ☐ Bill me ☐ Bill

Company Charge VISA ☐ MC # _____

Exp _____ ☐ Send sample issue here's \$3

Name & address _____

City State and ZIP _____

Mail to ACCESS PO Box 12847 Research Triangle Park,
NC 27709 Published by LEDS Publishing Co., Inc.

CP/M Exchange (Continued from page 45)

System now restricted to CP/M users only by use of a familiarity question.

Toronto, ON

Mississauga, Ontario HUG-RCP/M, (416) 273-3011
No A.L.D.S.; 1800-0600 weekdays, 24 hours weekends
110, 300, 450, 600, 710 baud
2+ Mb of files on 5 drives
Toronto Heath Users' Group.

South

Huntsville, AL

NACS/UAH RBBS/RCPM, (205) 895-6749, Don Wilkes
Call back, no A.L.D.S.; 24 hour operation
110, 300, 450, 600 baud
700 Kb files on 4 drives
Run for North Alabama Computer Society at University of Alabama; general CP/M software.

Louisville, KY

Louisville RBBS/RCPM, (502) 245-7811, Mike Jung
No call back; 0900-2100 weekdays, 24 hours weekends
300 baud; SPRINT, MCI
2.5 Mb of files on 5 drives
Heath/Zenith-based. Emphasis on BASIC software. Some HDOS stuff available for downloading.

Fort Mill, SC

Fort Mill RIBBS, (803) 547-6576, Bill Taylor
Up as of 08/15/81
No call back, no A.L.D.S.; 24 hour operation
300 and 1200 baud
3 Mb of files on 2 drives
Heath/Zenith-based with 212-compatible modem. The system carries ham stuff, general software, and on-line games. The fourth 1200-baud RCM.

DDJ

(Listing begins at right)

FORTH-79

Ver. 2 For your APPLE II/II+

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual	200 PG.	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
Both 13 & 16-sector format.	YES	_____
Multiple disk drives.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
LO-Res graphics.	YES	_____
80 column display capability	YES	_____
Z-80 CP/M Ver. 2.x & Northstar also available	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement option:		
Hi-Res turtle-graphics.	YES	_____
Floating-point mathematics.	YES	_____
Powerful package with own manual ,		
50 functions in all,		
AM9511 compatible.		
FORTH-79 V.2 (requires 48K & 1 disk drive)		\$ 99.95
ENHANCEMENT PACKAGE FOR V.2		
Floating point & Hi-Res turtle-graphics		\$ 49.95
COMBINATION PACKAGE		\$139.95
(CA res. add 6% tax: COD accepted)		

MicroMotion

12077 Wilshire Blvd. # 506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.



FORTH-79

Version 2 For Z-80, CP/M (1.4 & 2.x),
& NorthStar DOS Users

The complete professional software system, that meets ALL provisions of the FORTH-79 Standard (adopted Oct. 1980). Compare the many advanced features of FORTH-79 with the FORTH you are now using, or plan to buy!

FEATURES	OURS	OTHERS
79-Standard system gives source portability.	YES	_____
Professionally written tutorial & user manual.	200 PG.	_____
Screen editor with user-definable controls.	YES	_____
Macro-assembler with local labels.	YES	_____
Virtual memory.	YES	_____
BDOS, BIOS & console control functions (CP/M).	YES	_____
FORTH screen files use standard resident		
file format.	YES	_____
Double-number Standard & String extensions.	YES	_____
Upper/lower case keyboard input.	YES	_____
APPLE II/II+ version also available.	YES	_____
Affordable!	\$99.95	_____
Low cost enhancement options;		
Floating-point mathematics	YES	_____
Tutorial reference manual		
50 functions (AM9511 compatible format)		
Hi-Res turtle-graphics (NoStar Adv. only)	YES	_____
FORTH-79 V.2 (requires CP/M Ver. 2.x).		\$99.95
ENHANCEMENT PACKAGE FOR V.2:		
Floating point		\$ 49.95
COMBINATION PACKAGE (Base & Floating point)		\$139.95
(advantage users add \$49.95 for Hi-Res)		
(CA. res. add 6% tax; COD & dealer inquiries welcome)		

MicroMotion

12077 Wilshire Blvd. # 506
L.A., CA 90025 (213) 821-4340
Specify APPLE, CP/M or Northstar
Dealer inquiries invited.



Circle no. 19 on reader service card.

(Text begins on page 42)

```

IF      STDCPM
ASE     EQU    0
        ENDIF

```

BASE+5

(Continued on next page)

MBOOT.ASM

(Listing continued, text begins on page 42)

```

FCB EQU BASE+5CH
;
; ORG BASE+100H
;
LXI H,0
DAD SP
SHLD STACK
LXI SP,STACK
CALL INITADR
CALL ILPRT
DB 'MBOOT as of '
DB '11/5/80',CR,LF,0
LDA FCB+1
;
CPI
JNZ
CALL ILPRT
DB '++NO FILE NAME SPECIFIED++',CR,LF,0
JMP EXIT
;
CALL INITMOD
IN MODDATP
IN MODDATP
CALL ILPRT
DB 'CR,LF,TERMINAL MODE',CR,LF
DB 'CTL-E exits to CP/M, ESC starts file transfer',
DB CR,LF,0
;
CALL STAT
JZ TERM
CALL KEYIN
CPI EXITCHR
JZ EXIT
CPI ESC
JZ RCVFIL
OUT MODDATP
;
IF NOT DCH
IN MODCTLP
ENDIF
;
IF DCH
IN MODCTL2
ENDIF
;
ANI MODRCVB
CPI MODRCVR
JNZ TERM
IN MODDATP
ANI 7FH
CALL TYPE
JMP TERM
;
RCVFIL CALL ERASFIL
CALL MAKEFIL
CALL ILPRT
DB 'FILE OPEN, READY TO RECEIVE',CR,LF,0
;
RCVLP CALL RCVSECT
JC RCVBOT
CALL WRSECT

```

(Continued on top of page 48)

```

INCRSND
CALL SENDACK
RCVLP
;
RCVEDT
CALL WRBLOCK
CALL SENDACK
CALL CLOSFIL
CALL ERXIT
DB CR,LF,'TRANSFER COMPLETE$'
;
RCVSECT XRA A
STA A
ERRCT
;
RCVRPT B,10
RCV
RCVSECT
JC SOH
RCVSOH
A
RCVRPT
EDT
RZ
;
RCVSERR MVI B,1
CALL RCV
JNC RCVSECT
MVI A,NAK
SEND
ERRCT
A
ERRCT
ERLIM
CPI RCVRPT
JC
;
RCVSABT CALL CLOSFIL
CALL ERXIT
DB CR,LF,'++UNABLE TO RECEIVE BLOCK'
DB CR,LF,'++ABORTING++$'
;
RCVSOH MVI B,1
CALL RCV
RCVSECT
D,A
MVI B,1
CALL RCV
RCVSECT
D
RCVDATA
RCVSECT
A,D
RCVSND
C,0
H,BASE+80H
;
RCVCHR MVI B,1
CALL RCV
JC RCVSECT
MOV M,A
L
RCVCHR
D,C
MVI B,1
CALL

```

(Continued on page 49, column 1)

(Listing continued, text begins on page 42)

<---THIS RET MUST BE HERE

WTYPE

(Continued on column 3)

(Continued on column 3)

Circle no. 90 on reader service card.

16-BIT SOFTWARE TOOLBOX

by Ray Duncan

Late News for 8087 Fanciers

Intel recently dropped the price of the Intel 8087 numeric processor to \$230 in single-unit quantities. Some high-level software tools to exploit the 8087 are also beginning to appear. In addition to my own company's version of Forth which can use the 8087 co-processor, Lifeboat Associates is marketing the Lattice C Compiler which includes an 8087 runtime library, and rumor tells of a UCSD Pascal support file named 8087.PLS which is not yet officially announced.

16-Bit Tools, Installment # 2

The accompanying listing is an 8086/88 assembly language subroutine that returns the sine or cosine of an angle to four significant digits. The precision is adequate for most graphics applications, and the routines are implemented by a combination of deduction and table-lookup techniques that are extremely fast. The original version of these routines was coded in Forth by John James and placed in the public domain; interested

readers are referred to his article in *Forth Dimensions*, volume 4, no. 1.

Readers who also need a tangent function can derive it easily:

$$\tan(x) = \frac{\sin(x) * 10000}{\cos(x)}$$

But watch out: finding tangents in this way for angles greater than seventy-two degrees will lead to a divide overflow, which causes a hardware interrupt on the 8086/88.

A more precise (but slower) method of finding sine, cosine and tangent through a simple polynomial expansion will be presented in a later column.

Product Report: Microsoft RAMCard with RAMDrive

This integrated combination of hardware and software can be installed in any IBM Personal Computer, and will markedly enhance its operation. "RAMDrive" refers to a technique of mapping a disk directory and file structure onto a portion of the machine's random access memory.

The operating system drivers are modified so that the simulated disk can be accessed by application programs like any other peripheral device. Since there are no moving parts, data can be retrieved nearly instantaneously. As many as three RAMCards may be installed, yielding a maximum of 768 Kbytes of RAMDrive storage.

The package contains a memory expansion board of average-quality construction socketed for 256 Kbytes with parity, a diskette of utility programs and a sixty-page manual. The product is available in several versions and prices, depending on the amount of RAM initially supplied. The user can upgrade a partially filled card by simply plugging in more dynamic memory chips.

Installing the memory card in the IBM PC is quite simple, and can be done in a few minutes by anyone who can read and handle a screwdriver. The manual is clear and complete, and has plenty of helpful troubleshooting tips.

Next, you use a memory test provided on the Microsoft diskette to make sure

NOW - A POWERFUL Z80 CP/M¹ EDITOR FOR ONLY

\$25

That's right. WYLBUR² editors have been popular on IBM mainframes for over a decade. Micro-WYL implements all of the common WYLBUR commands with complete features for

collecting, inserting, deleting, replacing, and modifying text. Moving or copying text within a document. Copying text from external files. Global search and change operations. Listing control.

Marketing experts recommended a price of \$75-300

In fact Micro-WYL has been marketed at \$250/copy. So why are we selling it at \$25? We have cut our overhead to almost nothing - you are getting the product directly from its authors. We believe that there is an enormous market for high-quality, low-cost software. We intend to make a handsome profit by having thousands of satisfied customers.

We guarantee your satisfaction

If after reading the manual you feel that Micro-WYL is not for you, send it back with the sealed diskette unopened within 30 days. On the other hand, if you decide that it's as good as we believe, we authorize you to make copies of the manual and diskette for your friends at \$15 each. We'll put all of you on our mailing list for updates and enhancements.

If you feel that you must have proportional-spacing, justification, etc. there are good word processors in the \$75-500 range. But for most users, Micro-WYL represents an incredible bargain. It is easy to learn, easy to use, and incredibly powerful. **Send your check today - we'll ship within 48 hours of receipt.**

¹ CP/M is a registered trademark of Digital Research, Inc.

² WYLBUR is a registered trademark of The Board of Trustees of the Leland Stanford Junior University

Make your check out to:

Realworld Software, Inc.
913 S. Fourth St.
Suite 103
DeKalb, IL 60115

Amount:

\$25 plus \$2 for
postage and handling.
IL residents add 5%
for sales tax.

CRT type _____

☐ 8" SSSD

☐ ALTOS Series 5

☐ KAYPRO II 5"

☐ NEC 5"

☐ NorthStar 5" DD

☐ Apple/Softcard

Name _____

Address _____

City _____ State _____ Zip _____

the RAMCard is working properly. This program is a classic of its kind. It was obviously written by a programmer who was told to make the product "user-friendly," and who thought he could meet that objective with liberal use of reverse-video, blinking text, complicated screen-oriented presentations and, of course, using the special-function keys at any cost. For example, the user is asked to respond to one of the program's setup questions by pushing either the F9 function button or the space bar, neither of which are conveniently located or have the slightest mnemonic value. In spite of its veneer of razzle-dazzle, the program suffers from major design faults in the human interface that make it awkward to use, such as a lack of backspace capability should you be unfortunate enough to make an error in numeric entry. The program and documentation don't provide the smallest morsel of information about the testing process itself, and the only way to terminate the test once it is running is to reset the computer. This is not my idea of a classy program.

After you are convinced that the memory card is working correctly, you run the configuration program, which patches the PC-DOS operating system and makes the extra memory available as a simulated disk device. This is a simple procedure and requires only a few seconds — afterwards, you can copy the revised operating system to your various working disks. You might wonder what would prevent someone from pirating the configuration programs and using them with other companies' less-expensive memory cards. Never fear, Microsoft has provided against this eventuality: part of the necessary software is in a programmed read-only memory chip installed right on the RAMCard.

Using the RAMDrive capability can result in astonishing improvements in performance for programs which are disk I/O bound. For example, a simple Forth program to sequentially read through a 100 Kbyte file, accessing it as 1024-byte records, completed in 24.9 seconds using a regular 5¼-inch floppy disk drive but only required one second on the RAM-

Drive.

The RAMCard and RAMDrive package costs \$495 with 64Kbytes installed, and each additional 64Kbytes costs you \$200 from Microsoft up to a grand total of \$1095 for a complete 256Kbyte card. Here is where a little careful shopping can save you a bundle of money. I bought the minimum 64Kbyte card from Microsoft via Computerland, and obtained the twenty-seven dynamic RAM (type 4164) chips necessary to fill up the card for \$227 from BG Micro in Garland, Texas. Thus, I saved \$373 as compared to buying the RAM expansions from Microsoft.

Note that if you do not want the RAMDrive feature of this product, there are many memory cards available from reputable companies at considerably lower cost. But if you really need increased speed from your IBM PC, but aren't yet ready to take the plunge on a hard disk, then this product is great!

(Listing begins at right)

Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 530



the **FRIENDLY COMMUNICATIONS SOFTWARE** that has been **EASY TO USE** since 1978

- Auto Dial + Answer Turnkey package with BBS
- COMM Smart Terminal and File Transfer w/ Mainframe Protocols, CRC16 BiSync + Batch
- CONSOLX Remote System Access Controller
- Bulletin Board System w/Data File Manager
- Utilities included for KeyMacros + Sort Dir
- Fortran available for Mainframe BiSync
- Detailed User Manual Available For \$20
- CPU License \$150 Object or \$900 Source



HAWKEYE GRAFIX

Contact Your Local Dealer or Call or Write For Free Brochure
23914 Mobile, Canoga Park, Ca. 91307 U.S.A. • 213/348-7909

Circle no. 141 on reader service card.

8086 Trig

(Text begins on page 51)

```

title 'Trig Lookup Functions'
pagewidth 79

;
; Trig lookup functions for 8086/88
; adapted by Ray Duncan from a public
; domain FORTH routine written by John James.
;
; 'Trig' is a common routine used by 'sin'
; and 'cos' to reduce an angle into the
; range 0-90 degrees then extract the
; appropriate function value from the table.
;

0000 BRDB      trig:  mov     bx,ax    ;bx <- degrees
0002 83FB5A      cmp     bx,90      ;if > 90 degrees
0005 7E06      000D      jle     trig1 ;reduce the angle
0007 B1EBB400    sub     bx,180
000B F7DB      neg     bx
000D D1E3      trig1:  sal     bx,1    ;table index = 2*deg.
                                ;now extract value
000F 2E8BB73F00  mov     ax,sintbl [bx]
0014 C3      ret     ;back to caller

;
; Cosine lookup: add 90 degrees to
; argument and use sine lookup.
; call with: ax = degrees
; returns: ax = cosine * 10000
; other registers preserved.
;

0015 055A00      cos:  add     ax,90

;
; Sine lookup: reduce to angle in
; range 0-359 degrees, then call
; 'trig' to extract function value
; from table.
; call with: ax = degrees
; returns: ax = sine * 10000
; other registers preserved.
;

001B 52      sin:  push     dx        ;save registers
0019 53      push     bx
001A 99      cwd        ;deg -> double prec.
001B BB6801  mov     bx,360      ;reduce angle to
001E F7FB  idiv     bx        ;range 0-359 degrees
0020 B8C2      mov     ax,dx      ;let ax=remainder
0022 0BC0      or     ax,ax      ;is angle negative?
0024 7903      0029      jns     sin2 ;no, jump
0026 056801  add     ax,360      ;yes,make it positive
0029 3DB400  sin2:  cmp     ax,180 ;now reduce angle to
002C 7E0B      0039      jle     sin3 ;range 0-180 degrees
002E 2DB400  sin3:  sub     ax,180 ;angle was > 180 deg.
0031 E8C0FF  0000      call    trig ;look up function value

                                ;
                                ;
0034 F7DB      neg     ax
0036 E90300  003C      jmp     sin4
0039 E8C4FF  0000      call    trig ;angle was <= 180 deg
003C 5B      sin4:  pop     bx      ;restore registers
003D 5A      pop     dx
003E C3      ret     ;back to caller

;
; lookup table for trig functions
;
sintbl dw 0 ;0 degrees
dw 175 ;1
dw 349 ;2
dw 523 ;3
dw 698 ;4
dw 872 ;5
dw 1045 ;6
dw 1219 ;7
dw 1392 ;8
dw 1564 ;9
dw 1736 ;10
dw 1908 ;11
dw 2079 ;12
dw 2250 ;13
dw 2419 ;14
dw 2588 ;15
dw 2756 ;16
dw 2924 ;17
dw 3090 ;18
dw 3256 ;19
dw 3420 ;20

```

(Continued on next page)

Twenty-five Hours-A-Day

- Time Manager gTIME-2001 \$40
- Time Accounting/Reports gTIME-2101 \$80

Project Planning and Control

- PERT/CPM Utility gPERT-3001 \$40
- Project Scheduling/
Risk Analysis gRISK-4001 \$160
- Activity Selection Via
Multiple Objectives gPRIORITY-5001 \$160
- Project Report Generator gREPT-1001 \$40
- Time System-Project
System Interface gTIME-2201 \$40

People Interface Assistants

- People Networking
Assistant gPAN-1001 \$40
- Personal Address Disk/
Mail Labels gPAD-1001 \$40



GALACTIC

Software Publishers

2490 Channing Way, #5013

Berkeley, CA 94704, USA

(415) 845-5396

Circle no. 23 on reader service card.

USE FORTH!

Join the
FORTH Interest Group
and receive **FORTH**
Dimensions, a bi-
monthly magazine.

The best way to learn the FORTH language and to keep abreast of the latest FORTH developments is to join the FORTH Interest Group. . . over 2,500 members, worldwide. \$15.00 US, \$27.00 Foreign. Send check or money order to:

FORTH INTEREST GROUP
PO Box 1105
San Carlos, CA 94070

Circle no. 87 on reader service card.

8086 Trig

(Listing continued, text begins on page 51)

```
0069 000E      dw      3584      j21
006B A20E      dw      3746      j22
006D 430F      dw      3907      j23
006F E30F      dw      4067      j24
0071 B210      dw      4226      j25
0073 2011      dw      4384      j26
0075 BC11      dw      4540      j27
0077 5712      dw      4695      j28
0079 F012      dw      4848      j29
007B 8B13      dw      5000      j30
007D 1E14      dw      5150      j31
007F B314      dw      5299      j32
0081 4615      dw      5446      j33
0083 DB15      dw      5592      j34
0085 6B16      dw      5736      j35
0087 F616      dw      5878      j36
0089 B217      dw      6018      j37
008B 0D18      dw      6157      j38
008D 9518      dw      6293      j39
008F 1C19      dw      6428      j40
0091 A119      dw      6561      j41
0093 231A      dw      6691      j42

0095 A41A      dw      6820      j43
0097 231B      dw      6947      j44
0099 9F1B      dw      7071      j45
009B 191C      dw      7193      j46
009D 921C      dw      7314      j47
009F 071D      dw      7431      j48
00A1 7B1D      dw      7547      j49
00A3 EC1D      dw      7660      j50
00A5 5B1E      dw      7771      j51
00A7 CB1E      dw      7880      j52
00A9 321F      dw      7986      j53
```

```
00AB 9A1F      dw      8090      j54
00AD 0020      dw      8192      j55
00AF 6220      dw      8290      j56
00B1 C320      dw      8387      j57
00B3 2021      dw      8480      j58
00B5 7C21      dw      8572      j59
00B7 D421      dw      8660      j60
00B9 2A22      dw      8746      j61
00BB 7D22      dw      8829      j62
00BD CE22      dw      8910      j63
00BF 1C23      dw      8988      j64
00C1 6723      dw      9063      j65
00C3 AF23      dw      9135      j66
00C5 F523      dw      9205      j67
00C7 3B24      dw      9272      j68
00C9 7B24      dw      9336      j69
00CB B524      dw      9397      j70
00CD EF24      dw      9455      j71
00CF 2725      dw      9511      j72
00D1 5B25      dw      9563      j73
00D3 BD25      dw      9613      j74
00D5 BB25      dw      9659      j75
00D7 E725      dw      9703      j76
00D9 1026      dw      9744      j77
00DB 3526      dw      9781      j78
00DD 5B26      dw      9816      j79
00DF 7B26      dw      9848      j80
00E1 9526      dw      9877      j81
00E3 AF26      dw      9903      j82
00E5 C526      dw      9925      j83
00E7 D926      dw      9945      j84
00E9 EA26      dw      9962      j85
00EB FB26      dw      9976      j86
00ED 0227      dw      9986      j87
00EF 0A27      dw      9994      j88
00F1 0E27      dw      9998      j89
00F3 1027      dw      10000      j90 degrees
```

end

End Listing

Now You Can Develop 8086/8088 Code on Your 8080/Z-80 with Jonathan Mills' XASM86™ Cross-Assembler

Bridge the gap between the 8080/Z-80 & the powerful new 8086/8088 with XASM86. This highly acclaimed cross-assembler as described in the June 1981 & April 1982 issues of Dr. Dobb's Journal is available now

The XASM86 disk contains 15 files including:

- THE CROSS-ASSEMBLER
- MNEMONIC TABLE FOR XASM86
- THE XASM86 LOADER
- SOURCE FILE OF I/O INTERFACE TO CP/M
- DEMONSTRATION PROGRAM FILES
- A FILE COMPARE UTILITY

XASM86 features include:

- MATH SUPPORT Intel 8087 support included
- I/O Shipped with standard CP/M* I/O installed
- LABELS All characters significant (200 + possible)
- INCLUDE May nest included source files up to 4 deep
- ERROR HANDLING Over 100 error & warning messages
- COMMENTS Selective display on output listings
- SYMBOL TABLE Optionally displays, sorts & groups by usage

The above package including 100+ pages of documentation is yours for only \$200.00 (postage paid USA & Canada).

- XASM86 disk formats include standard 8" CP/M or CP/M on Micropolis
- Special Source Code Package available. Contact us for details.

Also: Jonathan Mills' original Micropolis MDOS source files available on Micropolis or standard 8" CP/M. Executable under Micropolis MDOS only. Price \$50.00.

RD SOFTWARE 1290 Monument St., Pacific Palisades, CA 90272 (213) 459-8119

CA residents add 6½% sales tax: *CP/M™ of Digital Research Corp.

Circle no. 244 on reader service card.

by Michael Wiesenber

A Potpourri of Good-Priced Hardware

The three-inch Winchesters are coming! One of the first is from SyQuest Technology. Their **SQ306** they claim is "the industry's first 3.9-inch (100mm) removable cartridge Winchester disk drive." SyQuest is aiming at the OEM market, but the price of \$400 per unit in large quantities means we can expect to see these five-megabyte Winchesters arrive reasonably priced. The cartridges will sell in quantity for \$30 each. The small size of this unit (1.625 inches tall) allows two SyQuest drives to fit into the same space as one 5.25-inch Winchester or mini-floppy drive. We should see these units integrated, probably stacked two at a time, into the next generation of portable computers. **Reader Service No. 306.**

Monitor the status of the "seven most important" (we are told) RS-232 lines with, what else?, **RS-232 Tester** from B & B Electronics. A female 25-pin connector at one end and a male connector at the other let you attach the Tester directly to the interface and leave it permanently in the line, with LEDs constantly displaying status, but not interfering with data transfer ability. \$39.95 postpaid. **Reader Service No. 316.**

The **SSB-MPF Speech Synthesizer Board** from Etronix provides a 400-word vocabulary for the MPF-1 Micro-Professor, the Z-80-based single-board computer-in-a-book described in a previous column. The board does its input/output with the Micro-Professor's keyboard and speaker, and features a 4K time clock and speech utility EPROM, two EPROM sockets that can be used to expand the vocabulary, adjustable pitch and volume controls, a power adaptor, all necessary connection accessories and a manual. \$129. **Reader Service No. 326.**

A Gallimaufry of Inexpensive Software

AUTODIFF is a file difference detector for CP/M by the Software Toolworks. Not only does this comparison utility list differences between files or produce a copy of the file with all changes flagged, it also reports insertions, deletions and changes, listing them in ASCII or hexadecimal format to a terminal, printer or disk file. Filters can be set to ignore non-printing characters or to display control characters. **AUTODIFF** can be used on most files and costs \$29.95 (plus \$2 postage and handling) on 5-inch disk for

Osborne I and Heath/Zenith, and 8-inch CP/M disk. **Reader Service No. 346.**

Disk-Edit is a screen-oriented disk editor from Supersoft for CP/M programs. It can call up files that, according to Supersoft, "are not even accessible with a normal text editor, then edit those files in either ASCII or hexadecimal notation." The program loads a one-kilobyte section of a disk (hard or floppy) into a buffer, and then displays a dual-view "window" into the buffer. On the left are the hexadecimal values of each byte; on the right, the ASCII. Change either, and the corresponding value changes instantly in the other. Simple commands move the cursor up, down, left, right, to the next page or screen, to the start of the file and so on. There are also string searches and many other functions. **Disk-Edit** has a "terminal definition package" and can be configured to most CP/M systems. \$100, or \$15 for the manual only. **Reader Service No. 356.**

You probably thought that a text editor for the IBM Personal Computer would be expensive, but **WINDOW**, a full-screen text editor program from Intellect Associates Inc., costs \$150. It uses all the screen and keyboard capabilities of the PC, including single-stroke editing commands with the function keys. It's not quite a word processor, but **WINDOW** does most of the things a word processor can, such as easily move the cursor, scroll in four directions, do global search and replace, insert and delete characters and lines, move, copy, split and join lines and edit text files larger than available memory. **WINDOW** runs under IBM-DOS, comes on 5.25-inch diskette, with documentation, and requires 64K, one drive, and a monochrome display and adapter. To turn **WINDOW** into a real word processor, add Intellect's **PCTEXT** for \$100, and get a text processing package that indents, centers, controls line spacing and margins, justifies, inserts headings and footings, numbers pages, underlines and merges documents. It, too, runs under IBM-DOS, requires 48K, one drive and a printer. And while we're talking about Intellect Associates, they offer for the same configuration a data management system, **DMS**, a self-prompting, menu-driven system with which users create data entry forms on the screen in user-specified formats, and then enter, retrieve, modify, correct and delete data with the forms and print reports. The sequential ASCII data base files can be ac-

cessed by any language. **DMS** is written in C88 (Intellect Associates' one-pass-compiler subset of C that costs \$250, generates compact 8088 machine code and comes with a linker), which makes it run, they say, very fast, and also makes good use of the PC's screen and keyboard features. It requires 48K, one drive (two recommended) and monochrome display adapter. **Reader Service No. 366.**

Cross-assemblers can cost a bundle, but here's one for the Zilog Z8 that costs \$150. **SYSTEM-Z8**, by Allen Ashley, for CP/M, includes a down-loader for the Zilog Z8 Development Module to transmit developed programs for in-circuit test. You get a macro-assembler with macro and conditional assembly and chaining, interactive editor/assembler, text editor, cross-reference generator, complete documentation and user support by mail or phone. Current **SYSTEM-Z8** owners can phone for a free update. **Reader Service No. 376.**

Ensign Software offers a whole stack of reasonably-priced software for the IBM PC. **String Sort** is a machine-language sort routine for BASIC programs that sorts 1000 variable-length strings in less than four seconds, and 5000 three-character strings in fourteen. \$24.95. **Electronic Disk** uses RAM as an "electronic disk drive" and printer spooler. The entire contents of a drive are dumped

Programmer's Aid Package For CP/M Users \$45.00

- ★ **Unix-flavored utilities** designed to make documentation and text processing easier
- ★ **Utilities include:** text formatter, "pretty" printer, file transliteration, word and character counting, file encryption/decryption and more - 9 programs in all
- ★ **For CP/M 2.2 8" single sided single density disk systems -48k RAM or more**
- ★ **Users Manual only: \$3.00** (applicable towards purchase)

Send check or money order to:

L.A. Software

6708 Melrose
Los Angeles
California 90038

California residents add sales tax
CP/M is a trademark of Digital Research
Unix is a trademark of Bell Telephone Labs
Copyright 1982 L.A. Software

Circle no. 4 on reader service card.

BDOS ERROR ON B:BAD SECTOR



Before disk errors ruin your work again order BADLIM.

- BADLIM assures the reliability of your CP/M computer.
- You can use your disks 10 times longer without losing your data AND your time.
- BADLIM checks thoroughly your disk marking all the blocks which have defective sectors. The operating system will know that those sectors should be skipped.
- BADLIM is the only program that gives protection for soft and hard errors.
- The first time BADLIM will list which files in your disk are on bad sectors, so you can take action to correct it.
- But thereafter the bad areas in your disk will be automatically by-passed.
- For CP/M 1.4 single density and for CP/M 2.xx of any format and density. It is a must for Winchester as the media cannot be replaced.

BADLIM cost only \$73. Whatever the reason you have to use a computer you need BADLIM. Contact your dealer or call us today:

BLAT R&D Corp., 8016 188th. St SW, Edmonds
WA 98020. Phone: [206] 771-1408

DEALER INQUIRIES INVITED.

BADLIM

Circle no. 45 on reader service card.

to the memory-resident electronic disk drive, so that disk access programs do superfast I/O. You can also use the computer for other tasks while printing large files. \$69.95. **ISAM Database** accesses up to seven data files simultaneously by full or partial keys using the *Indexed Sequential Access Method*, and is easily called from BASIC. It sorts 5000 records in twelve seconds. A demonstration of ISAM Database in the form of a complete name and address program is included in the \$69.95 price. **Cross Reference** gives a complete cross-reference listing of a BASIC program saved in either ASCII or binary format. The listing includes all variables alphabetically with all line number references, line numbers referenced by other statements and all line numbers with PRINT and LPRINT. \$24.95. All programs require one drive, DOS, disk BASIC, either monochrome or color/graphics and 48K (except Electronic Disk, 128K). **Reader Service No. 386.**

Central Point's real bargain is their **FILER** disk utility for drives of any number of tracks. It checks drive speed, tests the drive, copies a disk in thirty-five seconds; and has a file manager that catalogs or copies files, disks and DOS; deletes, locks and unlocks files; and changes the name and file type of the boot program, all for \$19.95. **Reader Service No. 406.**

DDJ

NAME

Plum Hall Inc

SYNOPSIS

High quality technical training. Our workshops are intense full-mastery courses, rigorously tested with many customers, and cost effective for your projects.

TRAINING

C Programming Workshop
Idris* Workshop
Advanced C Topics Seminar
UNIX* Workshop
Pascal Programming Workshop

Public courses: \$1000 per person. In-house courses: \$9000 for 15 people.

PUBLICATION

C Programming Standards and Guidelines: readability and portability of C code. \$20 per copy.

PLUM HALL

Spruce Ave and Orchard
Rural Route 2 Box 235P
Pleasantville, NJ 08232
609-927-3770

SOFTWARE

Plum Hall Interface Library (PHIL): printf, getchar, and other portable UNIX functions for systems using Whitesmiths compiler. \$350 for single-CPU source code.

*TRADEMARKS

Idris: Whitesmiths, Ltd.
UNIX: Bell Laboratories

FALL SCHEDULE

San Francisco
New York City
Plum Hall, NJ
Los Angeles
Boston

J ADA^{*} N U S

Question: **When can I use Ada?**

Answer: **NOW —
with JANUS.**

Ada is available now for your micro-computer. **JANUS** is a subset of Ada which includes those features sorely missed by programmers on micros. Here is a list to help you decide for yourself.

- + Modular Separate Compilation
- + Single and Double Precision Floating Point Numbers
- + Binary Coded Decimal (BCD) Fixed Point Numbers
- + Integer and Long Integer Numbers
- + Complete String Handling
- + Sequential and Random Access I/O
- + Full Dynamic Allocation and Deallocation
- + Friendly Error Handling
- + An Assembler for interfacing assembly routines
- + A Linker for combining modules
- + True native code is produced
- + ROMable, reentrant code
- + Run-time library source code
- + Low Cost. **JANUS** is more cost effective than any other comparable Ada package
- + Inexpensive Updates
- + No royalties for programs written in **JANUS**
- + No hassle customer service

*The language
that is based
on the past
but looks to
the uses of
the future.*

In short, all pluses. **JANUS** contains everything you need to do fast, structured program development in a micro environment.

JANUS is available for the CP/M, CP/M-86, and MS-DOS operating systems.

Now you too can take a step forward into the future on these computers:

8080/Z80 based systems: (All CP/M) Apple Softcard, North Star, Cromemco, Superbrain, TRS-80 Model II, and all CP/M 8" disk systems.

8086 based systems: IBM Personal Computer, Victor 9000, Seattle Computer System II, Tecmar, Lomas Data Products, and all CP/M 8" disk systems.

8080 or Z80, CP/M (requires 56K memory) — \$300.00

8086/8088, CP/M-86 or MS-DOS (requires 96K memory) — \$400.00

CP/M, CP/M-86, MP/M-86 are trademarks of Digital Research, Inc.
* ADA is a trademark of the U.S. Department of Defense
MS-DOS is a trademark of Microsoft

86-DOS is a trademark of Seattle Computer Products
SB-86 is a trademark of Lifeboat Associates
Apple Softcard is a trademark of Microsoft, Inc.

©Copyright 1982 RR Software

RR SOFTWARE

specialists in state of the art programming

P.O. BOX 1512 MADISON, WISCONSIN 53701

(608) 244-6436

Circle no. 12 on reader service card.

well, but we'd rather see some evidence. Surely somebody out there has a PC, a suspicious mind, experience with the typical output of translator programs, and a few hours to spare. The MSDOS command DEBUG will disassemble storage. Use Ctrl-PrtSc to start a log of its output on the printer.

Actually, there are very good reasons why Microsoft would translate an existing interpreter rather than write a new one. Their MBASIC is a mature, well-debugged piece of code. Translation would let them produce a reliable product quickly and cheaply. Reliability, quick development, and low cost are three qualities that are conspicuously absent when you build a large piece of software from scratch. One can see how the "technical" consideration of execution speed could easily take a back seat to these desirable "management" criteria....

Burke Smith of Shawnee Mission, KS, has found a bug in that mature product, MBASIC. It only seems to surface in version 5.1 of BASIC-80, and in IBM version D1.00. Try this sequence:

```
LET A#=0
LET A#=-A#
PRINT INT(A#),FIX(A#)
```

Smith claims that in the noted versions (but not in the more common versions 4.51 and 5.2) two answers of -1 will be displayed, "rather than the zeroes that any reasonable person would expect."

Smith goes on to blame the problem on Microsoft's choice of binary floating-point rather than decimal float using a BCD representation. We think that's a red herring. We've seen a lot of misguided complaints along this line; the writers seem to think that there is something peculiar about the binary number system that makes it more prone to truncation errors ("round-off errors") than decimal is. Tain't so; a digit is a digit, no matter what your radix. The problem is the limited precision of the representation, not the number base. A decimal representation of a real number is just as prone to truncation errors as a binary, octal or hexadecimal one is.

There are standard techniques that will mitigate the effect of truncation errors. Hardware designers can incorporate an extra digit of precision in the temporary results held in machine registers, and round any bits that turn up in such

a "guard digit" back into the result before storing the result. Few people remember now, but the initial design for the IBM 360 line lacked guard digits in the floating-point ALU; the first machines had them added in the field as engineering changes. Good scientific calculators get the same effect by carrying several low-order digits that don't show in the display. MBASIC could have achieved the same effect in software; we don't know if it tries to.

A decimal representation is better in just one way: it will get truncation errors on the numbers that people expect it to. Nobody is too surprised when the expression $((1/3)*3)$ does not yield a result of 1.0; most people can picture the string 0.333333... being truncated and multiplied by 3 to yield 0.999999. The trouble with binary (or any other radix) is that some fractions that have finite represen-

tations in decimal become repeating "decimals" in another radix (and vice versa). When the system converts such a constant to its internal radix, it loses some precision before any calculations are done at all. For commercial arithmetic, the only answer is to use a fixed-point decimal representation like the ones available in PL/I and COBOL compilers. Floating-point arithmetic should *never* be used for commercial calculations, no matter what radix it is based on.

John Palmer of Boonville, CA, is grumpy about his new CP/M-86 system, which he procured from Godbout along with the Godbout 8085/8088 dual processor board. "The Godbout hardware is very good," he writes. "If you buy all the stuff from Godbout it will run." However, problems arose when he had to modify the Godbout BIOS. The first problem was that "the BIOS is very hard

SUBMIT.COM

```
; A patch to SUBMIT.COM which forces it to open an existing
; $$$SUB file for appending, rather than erasing it. The
; CCP reads that file in reverse order. Thus a SUBMIT within
; a submitted file will "push" new records onto the end of the
; file, from whence the CCP will "pop" them. The result is
; correct execution of nested SUBMIT commands.
;
subfcb equ 05BBh ; FCB for $$$SUB
BDOS equ 0005h
OPEN equ 0211H ; open-file subroutine of SUBMIT
;
org 022Dh ; the erase-file routine of SUBMIT
;
; It would seem that this location is called when an attempted
; open of $$$SUB succeeds, showing that the file exists.
; Previously, this code would have erased the file.
;
ops1 lda subfcb+15 ; open ok if extent not full
ral ; extent full if record count=>80h
rnc ; exit if count < 80h
lxi h,subfcb+12 ; extent number in FCB
inr m ; try the next extent
; Code to invoke or re-invoke the create-next-extent routine
; of SUBMIT--which will call "ops1" above if the extent exists?
ops lxi d,subfcb ; open first (next?) extent
jmp create
;
; Routine to create the $$$SUB file
;
create org 025Dh
call OPEN ; test $$$SUB by opening it?
inr a ; FFh -> 00h if open fails
jnz ops1 ; do above code if file exists
lxi d,subfcb ; file does not exist, make it
mvi c,22
call BDOS
adi 01h ; set carry flag if retcode=FFh
ret
;
; Replacement code for original logic to open $$$SUB
;
org 04FEh
call ops ; open last extent of $$$SUB
jc 0517h ; error if open failed
subfcb+15 ; use the FCB "record count"
sta subfcb+32 ; ..to set the "current record"
jmp 051Dh ; continue with original code?
;
end
```


to modify without the Sorcim anti-Intel ACT assembler." That would make us grumpy, too. There is no excuse for distributing a BIOS that requires an assembler that isn't part of the distributed system. "The source code for the BIOS is on the diskette," Palmer reports, "but you cannot use it until you buy Sorcim's ACT-86 for \$175." Palmer thinks this is a tacky way of selling assemblers, and we agree.

Passing the Hat

We're scraping the bottom of the mailbag again, readers. You've all been very flattering in your praise of this column, but you aren't *contributing*. We need questions, like Knipp's. Pitfalls and Warnings, like Palmer's. Bugs, like Barker's. Patches, like Pasky's. Discoveries, like Hammond's. Puzzles. Confusions. Grumps. Cheap software. *Material!* Please?



Reader Ballot

Vote for your favorite feature/article.
Circle Reader Service No. 526

Letters (Continued from page 6)

by *DDJ* readers. First, however, I attacked its primary problem: it's *slow*.

I studied the code for a while, and decided to change the global symbol table from a linearly searched list to a hash table. I also put the macros in the global symbol table, with pointers into the array of macro definitions. Since the original code was well-structured, changing algorithms was not difficult.

I changed "numglbs" from 300 to 512 (a power of two), "SYMTBSZ" from 5040 to 8008, and introduced "#define MASKGLBS 511". I introduced "#define MACRO 5", giving a new possible entry for "ident". Since I'm putting the macro names into the global symbol table rather than the macro table, I changed "macqsize" from 1000 to 500. "macptr" has to be initialized to 1 instead of 0. (Initializing it to 0 causes a subtle bug affecting only the first macro. I'll leave the details as an exercise for the student.)

I added the following initializing code to main():

```
glbptr=STARTGLB;
while(glbptr<ENDGLB){
    *glbptr=0;
    glbptr=glbptr+SYMSIZ;
}
glbptr=STARTGLB+SYMSIZ*5;
/* clear global symbols
*/
```

"dumpglbs()" needed the declaration "int i;", and the top of the loop changed to

```
i=NUMGLBS;
while(i--){ /* 6/19/82 jrvz */
    if(*cptr){
        if((cptr[ident]!=function)
            &(cptr[ident]!=MACRO))
            /* do if anything but
            function or macro
            jrvz 6/19/82 */
```

I introduced a simple hash function:

```
hash(sname)
char *sname;
{
    int h,c;
    h=*sname;
    while(c=*(++sname)) h=(h<1)+c;
    return h;
}
```

MICROSTAT® - Release 3.0

MICROSTAT® + baZic® = PERFORMANCE

The best just got better! MICROSTAT has been the leader in the statistics field for microcomputers since 1979, and the new release 3.0 outperforms and is noticeably faster than previous versions. Just a few of the features include:

GREATER ACCURACY

BCD with up to 14 digit precision;

PROGRAM ENHANCEMENTS

Missing data capabilities and many more;

FASTER EXECUTION

Calculation time greatly reduced;

DYNAMIC FILE ALLOCATION

Data can be inserted, added, or deleted;

SPECIAL PRICE:

For a limited time get MICROSTAT plus baZic complete with program disk and documentation for each for \$395.00, save \$50.00!

The MICROSTAT - baZic version requires: a Z80 CPU, CP/M™ and 48K of memory. Available formats: 8" SD disk or 5¼" North Star only. Check with your dealer for other formats. Also available for: Microsoft's Basic-80™, North Star DOS and IBM. For more information, call or write:

ECOSOFT INC.

P.O. Box 68602
Indianapolis, IN 46268-0602
(317) 255-6476



MICROSTAT is a registered trademark of ECOSOFT, INC.
baZic is a registered trademark of MICROMIKES, INC.
CP/M is a registered trademark of DIGITAL RESEARCH
Basic-80 is a registered trademark of MICROSOFT

Circle no. 51 on reader service card.



STARSIDE
ENGINEERING

FOUR STARS FOR CP/M

RUNIC 80

Your introduction to the world of threaded-code interpreters. A complete programming language which sets you down easy in the realm of stacks, words, dictionaries, and RPN. Includes 60 page user guide, reference card, and numerous sample programs.

Order RN80 for CPIM in 8" SD, 5" Apple][, 5" Heath Org 0 --- \$49.95

BLAZE/LIB

Get a head start on your next Pascal/MT+ application with our library of utility routines. Number/string conversions, error trapping, data entry routines, directory search, fast rename, many more. Included is TLIB, a linkable module defining cursor control for most popular terminal devices including Soroc, Heath, Lear-Siegler, Xerox, ISC, SOL, and TRS80-2. Lets you run a single application on many terminals without changing the code!

Order BLLB for CPIM in 8" SD, 5" Apple][--- ERL \$75 / Source \$200

BLAZE/IO

Cut the size and increase the speed of your Pascal/MT+ code with our assembly-code implementation of all MT+ file I/O functions and procedures. (MT+'s file I/O is written in Pascal.) Includes a text file APPEND procedure.

Order BLIO for CPIM in 8" SD, 5" Apple][--- ERL \$75

PHONEDEX I

An "electronic little black book" which will store, search, list, and (with an autodial modem) dial telephone numbers for you. Includes a terminal emulator for communicating with CBB5 and timesharing systems, user-defined screen prompts, mailing label print, and 3" X5" address book page print functions. Currently supports Hayes Smartmodem; PMMI soon. Pascal/MT+ source for modem/dialer code is included. 2 SD 8" drives and 48K required.

Order PDX1 for CPIM in 8" SD, 8" TRS80-2 DD --- \$49.95

Any Starside Engineering user guide may be purchased for \$15, creditable against full package. We also offer a growing line of software for the IBM Personal Computer.

Terms: add \$3 P&H. NYS residents add 7% sales tax. MC-VISA accepted; evenings best for phone orders. CPIM, Pascal/MT+, Apple][, Smartmodem, and TRS80 are registered trademarks.



Starside Engineering
PO Box 18306
Rochester NY 14618
(716) 461-1027



Circle no. 33 on reader service card.

Something New

When you can't find your problem, let ACTIVE TRACE show it to you! See inside your program as it's working! Just as important, see inside your program when it's *not* quite working!

New to Basic? ACTIVE TRACE will let you see what Basic does as it does it! ACTIVE TRACE displays the line number, name, and current value of the variables and functions you choose, as they are encountered in program flow.

Something Old

Though less exciting than harnessing the power and speed of your computer to find mistakes, using your computer to avoid mistakes in the first place is equally valuable. Cross-reference utilities have been around for a long time. Most programmers would not attempt to work without them, and we don't know why they have not become more well known and understood among Basic programmers and educators. ACTIVE TRACE produces complete cross-reference maps and explains their use and importance.

Active Trace

If you have great intuition and are well-disciplined, then you'll want ACTIVE TRACE. But if you're like the rest of us, you need ACTIVE TRACE to:

- Understand and modify programs you did not write
- Improve your programming skills
- Minimize program development time

\$125.00

complete with primer to help you use ACTIVE TRACE to improve your programming.

Why pay more for cross-reference utilities alone when you can have ACTIVE TRACE, the new easy to use programming environment for the Microsoft family of Basic interpreters.

SOFTWARE
SOFTWARE
DIGITAL MARKETING
DIGITAL MARKETING™



DIGITAL MARKETING CORPORATION

2670 CHERRY LANE • WALNUT CREEK • CALIFORNIA • 94596
(415) 938-2880 • Telex 17-1852 (DIGMKTG WNCK)

ACTIVE TRACE is a Trademark of The Data Works.

Circle no. 21 on reader service card.

I then rewrote the following routines where boxes indicate changes):

```
findglb(sname)
/* cptr is set to entry if found,
or appropriate empty slot if not */
char *sname;
{
  int h;
  h=hash(sname)&MASKGLBS;
  cptr=STARTGLB+h*SYMSIZ;
  while(0==
  astreq(sname,cptr,namemax)){
    if(*cptr==0) return 0;
    cptr=cptr+SYMSIZ;
    if(cptr==ENDGLB)cptr=STARTGLB;
  }
  return cptr;
}
```

```
addglb(sname,id,typ,value)
char *sname,id,typ;
int value;
{
  char *ptr;
  if(findglb(sname))return cptr;
  if(glbptr>=ENDGLB)
    { error("global symbol table
    overflow");
      return 0;
    }
  ptr=cptr;
  while(an(*ptr++ = *sname++));
  /* copy name */
  cptr[ident]=id;
  cptr[type]=typ;
  cptr[storage]=statik;
  cptr[offset]=value;
  cptr[offset+1]=value>>8;
  glbptr=glbptr+SYMSIZ;
  return cptr;
}
```

```
addmac()
{ char sname[namesize];

  if(symname(sname)==0)
    {illname();
      kill();
      return;
    }
  addglb(sname,MACRO,0,macptr);
  while(ch()==" " | ch()==9)
    gch();
  while(putmac(gch()));
  if(macptr>=macmax)
    error("macro table full");
}
```

```
findmac(sname)
/* rewritten 6/19/82 jrvz */
char *sname;
{
  if((findglb(sname)!=0)&
  (cptr[ident]==MACRO))
    {return((cptr[offset]&255)+
    (cptr[offset+1]<<8));
    }
  return 0;
}
```

As a result of these changes, the compiler speed has more than doubled. I also enlarged the disk buffers from 128 bytes to

512 bytes. That sped compilation up a further 20%, to about 220 lines/minute on my 2.5 MHz Z-80.

Has anyone installed floating-point variables? Does anyone have a C interpreter, that would let me debug my code quickly? (Note that Tiny-c accepts a different language.)

Keep the Small-C articles coming!

Yours,
James R. Van Zandt
26 Shelton St.
Nashua, NH 03062

Editor's Note: Readers who are following the developments in Small-C, which originated with Ron Cain's Small-C compiler, should keep a close eye on DDJ in the coming months. We have a number of exciting articles and listings scheduled.

DDJ

A Professional Quality Z80/8080 Disassembler

REVAS Version 3

Uses either ZILOG or 8080 mnemonics
Includes UNDOCUMENTED Z80 opcodes
Handles both BYTE & WORD data
Disassembles object code up to 64K long!
Lets you insert COMMENTS in the disassembly!

A powerful command set gives you:

INTERACTIVE disassembly
Command Strings & Macros
On-line HELP
Calculations in ANY Number Base!
Flexible file and I/O control
All the functions of REVAS V2.5

REVAS:

Is fully supported with low cost user updates. Runs in a Z80 CPU under CP/M
Is normally supplied on SSD 8" diskette
Revass V 3...\$90.00 Manual only...\$15.00
California Residents add 6% sales tax

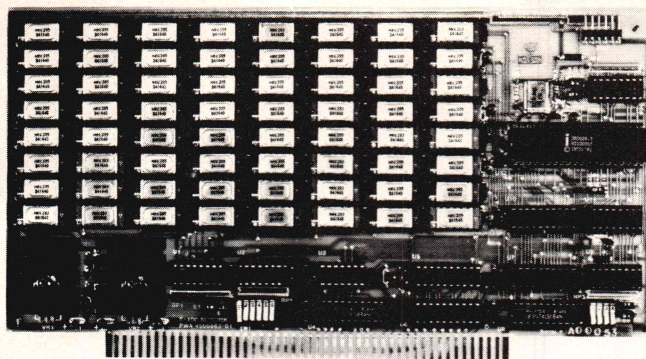
REVASCO

6032 Charlton Ave., Los Angeles, CA, 90056
(213) 649-3575

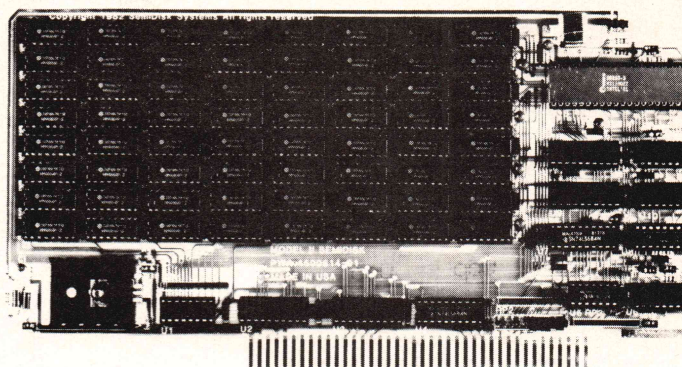
* CP/M is a Trademark of Digital Research, Inc.

Circle no. 84 on reader service card.

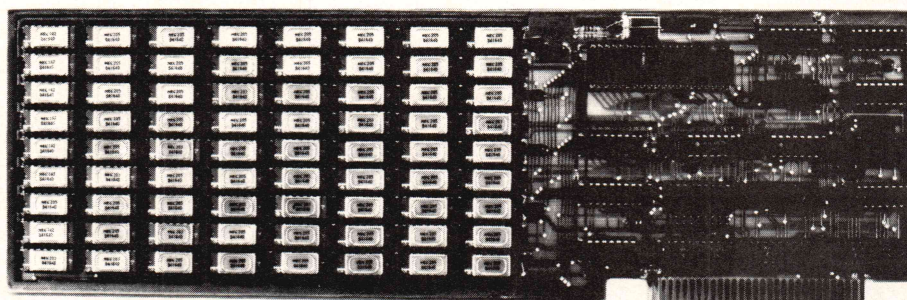
A FULL LINE OF SEMIDISKS



S-100



TRS 80 Model 2



IBM Personal Computer

Do you use your computer? Or does your computer "use" you? Face it, if you're using floppies, your time is being wasted. Because a floppy is an inefficient random access storage device. Each time the processor wants to transfer data, it has to wait an eternity for the disk to rotate and the head to move.

So what do you do? Get a SemiDisk, quick. It's a large capacity semiconductor memory board that is driven by software to operate like a disk drive. Without all the waiting. Do everything you'd do on a floppy or hard disk, with no modifications to your software or hardware. Two board sizes are available: 512K and 1 Megabyte. (the highest density microcomputer memory board in the world) And you can put up to 8 megabytes in a system by adding more storage boards.

What do you need to use it? Just an S-100 system with CP/M 2.2. Or a TRS-80 Model 2 system with CP/M 2.2. Or an IBM Personal Computer. That's it. No special processors, DMA, I/O, or disk controllers are required. Plug it in and run the installation program, and you're on your way. Fast! Even better, we supply full source code to the driver software, in case you'd like to do your own interfacing.

Best of all, the SemiDisk's price won't warp your wallet. Compare specs, cost/megabyte, storage capacity, and compatibility with the competition. You'll see that the SemiDisk is a disk emulator truly worthy of the name. SemiDisk has battery-backup capability, too.

Consider our limited warranty: A full year, covering all parts and labor. Consider our liberal 15 day return policy. Price? \$1995 for 512K byte SemiDisk, \$2995 for 1 Megabyte SemiDisk. Both from stock. \$10.00 for manual. VISA, Mastercard, COD orders accepted. Dealer and OEM inquiries welcomed. (Specify system type and disk format when ordering.)

Someday, you'll get a SemiDisk.

Until then, you'll just have to wait.

**SemiDisk
SYSTEMS**

P.O. Box GG
Beaverton, OR 97075

(503)-642-3100



Call (503)-646-5510 for CBBS®/NW, a Semi-Disk-equipped computer bulletin board.
SemiDisk trademark of SemiDisk Systems; TRS-80 trademark of Radio Shack

THINKING MAN'S GAMES FOR CP/M

ADVENTURE

The classic computer game, complete as implemented on the DEC-10. Explore the colossal caves—over a hundred rooms! Cast magic! Steal treasures! Run from dragons! Absolutely the fastest running version available for a microcomputer. **\$29.95**

OTHELLO

This ancient game of tactical skill is played on an 8 by 8 board. Object is to occupy the most squares at the end of the game by bracketing and "flipping" your opponent's men. Nine levels of play. **\$24.95**

GOMOKU

Playing on a 15 by 15 board, the first player to get five in a row wins this exciting game of tactical skill. Complete I/O features include game display, take back, autoplay, referee, set-up and problem modes. Can do a 5-ply search in tournament time. **\$24.95**



Indicate disk size and system type. 8" shipped single density; for 5 1/4" indicate soft or hard sector, single or double density. Add \$2 shipping on all orders. California residents add 6% tax.



CALIFORNIA DIGITAL ENGINEERING
P.O. BOX 526 ★ HOLLYWOOD, CA 90028

Circle no. 249 on reader service card.

LANGUAGE FOR TELECOMMUNICATIONS?

DTRANS Develops Transportable Applications in Real Time — and in Very Little Time!

- Query a Database
- Control Multiple Files
- Implement Electronic Mail
- Develop computer-to-computer Dialogs
- **DTRANS** isolates you from the host environment because **DTINSTAL DOES IT ALL!**

DTRANS also incorporates a terminal program with string recognition, software interface control, input filtering, block transfers, date time stamping, scroll through buffer, fast single key commands, buffered printing. **DTRANS TAMES HOSTILE REMOTE SYSTEMS.**

DTRANS is a unified concept for the programmer/developer/end user. Also available in turnkey configurations.

Order COD or send check or money order. Specify format: 8" or 5" single density or Northstar. OEM inquiries invited. For more information, write or call:

\$60 includes shipping in U.S.A. PA residents add 6% sales tax.

AUTOMATA DESIGN ASSOCIATES

1570 Arran Way Dresher, PA 19025 (215) 646-4894

ADVERTISERS INDEX

Reader Service No.	Advertiser	Page No.
183	Automata Design, Inc.	62
42	BDS C Users' Group.	38
45	Blat R&D Corp.	56
249	California Digital Engineering	62
255	Chromod Associates.	40
102	Code Works, The.	45
18	Computer Innovations	42
120	Compuview Products, Inc..	10, 11
232	CP/M Review	41
21	Digital Marketing.	60
233	Digital Research	Cover IV
90	D&W Digital.	50
51	Ecosoft, Inc..	59
258	Elliam Associates.	7
138	Epson America, Inc..	Cover III
87	FORTH Interest Group	53
23	Galactic Software	52
141	Hawkeye Grafix	52
273	Hazelwood Computer Systems.	15
226	Intellect Associates	25
261	JRT Systems	9
4	L.A. Software.	55
6	Laboratory Microsystems	Cover II
270	LEDS Publishing Co.	45
99	Manx Software	13
19	MicroMotion	46
81	Mountain View Press	12
195	PC/Quantum Software	5
293	Plum Hall Inc..	56
244	RD Software	54
93	Realworld Software.	51
84	Revasco.	60
12	RR Software	57
168	Semidisk Systems	61
299	Solution Technology, Inc..	26
33	Starside Engineering.	59
264	System/Z	35
72	Tiny C Associates	35
223	Transaction Storage Systems	3
276	Victor Business Systems	8
180	Vandata.	39
39	Western Wares.	42



Try this with an ordinary computer.

Epson.

The new Epson HX-20 is no ordinary computer. Not by a long shot. It's the world's only Notebook Computer with the power of a desktop and the portability of a handheld.

So you can do serious computing, data processing, even word processing. Anytime. Anywhere.

To start with the HX-20 has 16K RAM (optionally expandable to 32K), 32K ROM (optionally expandable to 64K), RS-232C and serial interfaces, a full-size ASCII keyboard, a built-in micro-printer with dot addressable graphics, a scrollable LCD screen, five programmable function keys, and... well, that's just the beginning.

The HX-20 is small enough to tuck inside a briefcase or under your arm. It runs on internal power for 50-plus hours and recharges in eight. It lets you interface with peripherals like MX Series printers, the CX-20 battery-powered acoustic coupler, a barcode reader, and audio cassette. And you can even get it with options like a micro-

cassette drive, ROM cartridge, floppy disk and display controller.

Now, prepare to have your mind boggled by one more feature: the price. The Epson HX-20 Portable Notebook Computer retails for less than \$800. That's right — less. Which means it's just right for students, businesspeople, kids — anybody who's looking for an affordable way into serious computing.

Powerful. Portable. Affordable. The HX-20 is just what you'd expect from Epson.

The extraordinary.



EPSON
EPSON AMERICA, INC.
COMPUTER PRODUCTS DIVISION

3415 Kashiwa Street • Torrance, California 90505 • (213) 539-9140

Circle no. 138 on reader service card.

CB80

Ultra *FAST* BASIC Compiler

CB80™ Compiler System's new native code Basic compiler, offers maximum speed and flexibility in creating applications to solve today's business problems.

CBASIC™ compatible: As an addition to the CBASIC family, CB80 has all the features of CBASIC (14 digit accuracy, long variable names, stream and record I/O, multiple line functions) plus these extras:

- Relocatable machine code • 32K byte strings
- Nested IF statements • ON ERROR GOTO
- Variable type declarations • CALL statement with parameters • EXTERNAL and PUBLIC functions
- Local variables in functions • Alphanumeric labels • Record LOCK and UNLOCK

Expand your versatility. CB80 includes our LK80™ linker. It allows you to create programs in separate modules and easily combine them. Powerful CHAINING capabilities, multiple library scanning, and easy linkage to assembly routines, are all part of LK80.

CP/M® and MP/M II™ compatible: CB80 supports the popular CP/M and MP/M II operating systems. CB80's record LOCK and UNLOCK, combined with its superior speed makes it a natural for multiuser environments.

Increase productivity and profits. Faster execution boosts system throughput and maximizes your computer's resources — an essential user feature. Coupled with reduced programming time, CB80 definitely improves your bottom line benefits.

For your free CB80 brochure and licensing details, call us at (408) 649-3896, or write us today.



DIGITAL RESEARCH®

P.O. Box 579, Pacific Grove, California 93950

Europe: Vector, Int'l., Leuven, Belgium, 32(16)202496

Far East: Microsoft Assoc., Tokyo, Japan, 03-403-2120

CBASIC™

Solving complex business problems with BASIC simplicity

CB80, CBASIC, LK80 and MP/M II are trademarks of Digital Research. CP/M is a registered trademark of Digital Research. © Copyright 1981 Digital Research